YOUR COMPUTER *Contents*

Ghost hunting: page 50.

ZX-81 software survey: page 42.

First Byters get even younger: page 38.

Sub hunt: page 78.

Cover photograph by Stephen Oliver.

IF YOU WANT PROFESSIONAL DEALER THERE'S A NEW

COMPUTERS FOR ALL Authorised Dealer!

From now on, you won't have to look far for a totally professional home computer dealer. Just look for the name that says it all. **COMPUTERS FOR ALL** It's the newest and most exciting network of Micro dealers in the U.K. The ideal places to buy, browse, seek advice and after-sales service . . . professional service! Every COMPUTERS FOR ALL dealer is dedicated to giving you, the customer exactly what you have alway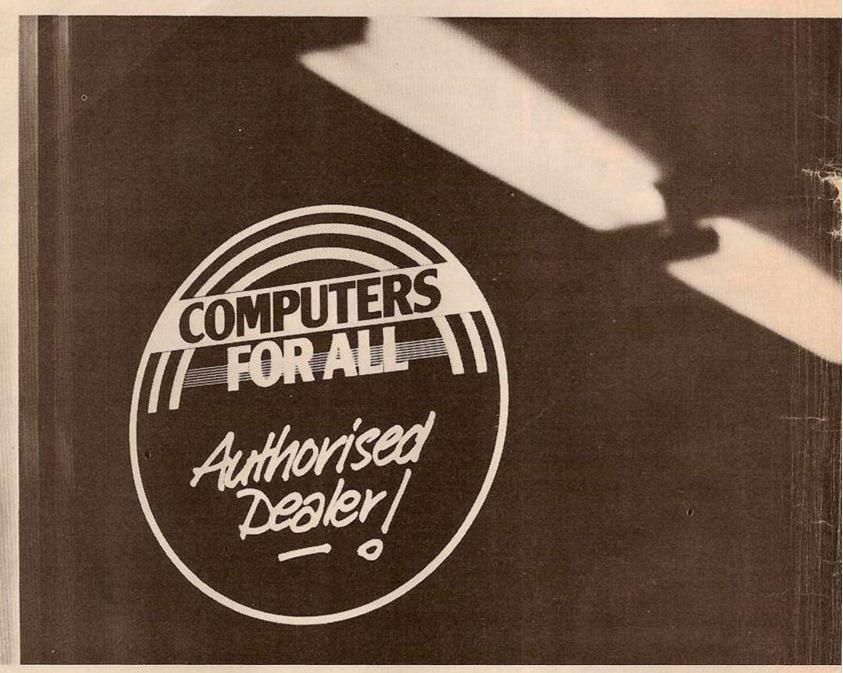s wanted . . . a shop or store that knows about computers, capable of helping you sort out what's right for your needs; expertise in giving sound advice, knowledgeable in technical matters. If you've ever bought from the usual retailer who just stocks and sells, you'll appreciate the difference! A COMPUTERS FOR ALL dealer is a shop where people matter . . . why not call in at your local one today. You'll get a friendly welcome!

You'll find a complete spectrum of software at W. H. Smith.
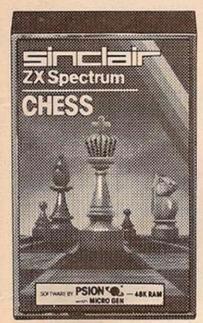
The range of software available for the Sinclair Spectrum is to say the least, extensive.

So every month at W. H. Smith, we review the range with particular reference to presentation, challenge, excitement and value for money. And because we're quite severe it's a difficult job finally to arrive at a Top Ten. But as you can see, not impossible.

You don't have to take our word for it.

At W. H. Smith we stock the widest selection of Spectrum Software, so you can judge for yourself.

Title	Producer	K RAM	Price
Arcade Games			
Meteor Storm	Quicksilva	16	£4.95
Space Intruders	Quicksilva	16	£4.95
Time Gate	Quicksilva	48	£6.95
Gulpman	Campbell	16	£4.95
Spectral Invaders	Bug-Byte	16	£5.00
Meteoroids	Softek	16	£4.95
Planetoids	Sinclair	16	£4.95
Hungry Horace	Sinclair	16	£5.95
Space Raiders	Sinclair	16	£4.95
Mazeman	Abersoft	16	£4.95
Nightflight	Hewson	48	£5.95
Ground Force Zero	Titan	16	£5.00
Caterpillar	CDS	16	£5.95
Leapfrog	CDS	16	£5.95
Gobble-A-Ghost	CDS	16	£5.95
Centi-Bug	DK Tronics	16	£4.95
Cruising	Sunshine	16	£4.95
Blind Alley	Sunshine	16	£4.95
Derby Day	Computer Rentals	48	£5.95
Jackpot	Computer Rentals	48	£4.95
Escape	New Generation	16	£4.95
Gobbleman	Artic	16	£4.95
Galaxians	Artic	16	£4.95
Invasion Force	Artic	16	£4.95
Sentinal	Abacus	16	£4.95
Cyber Rats	Silversoft	16	£5.95

Make much more of your computer!

16k RAM Cartridge for VIC-20
Non switchable

£35

while stocks last

More sound from your ZX Spectrum!

Echo

Not only more sound, but better sound and a wide range of other facilities!
Control Volume, and adjust tone of sound!
Load and Save without switching leads!
Audible cue facility for tape programs!
DIN compatibility!
No additional power supply needed!
Attractively cased — looks good
— SOUNDS GOOD!
Only £23.50

More memory for your Jupiter Ace

Pacer

New

The uniquely expandable 16K RAM pack
Similar concept to ZX-PANDA but for the incredible Jupiter Ace.
Attractive, solidly built 16K RAM pack with the facility of expanding to 32K by plug-in module.
For more power to faster FORTH ... you need a PACER!
16K Expandable RAM £29.95
16K Expansion Module. £19.95

More memory for your VIC-20

Vixen RAM Cartridge for the VIC-20

Switchable between 16K or 8K + 3K.
Gives you the option of full 16K RAM or 8K and 3K RAM in one package. When added to a Standard VIC-20 gives 16384 bytes of extra memory in memory blocks 1 and 2 or 3092 bytes of extra memory into the 3K memory block AND 8192 bytes of extra memory switchable between memory blocks 1 and 3.
Fully compatible with available motherboards/modules.
Simply plugs into the rear expansion port of computer.
No re-addressing of existing BASIC programs needed. **Only £44.95**

Tandem

Expandable Expansion System for the VIC-20
Gives 4 expansion slots for VIC-20 cartridges.
Custom-designed case. Plugs directly into computer. Further expanded by using TANDEM System!
ROM socket for expansion.
No extra power supply needed.
Only £34.95

To: Stonechip Electronics, Unit 9, The Brook Industrial Estate, Deadbrook Lane, Aldershot, Hants. Telephone: (0252) 318260.
Please forward me the following products:

. . . .

. . . .

All prices are inclusive of VAT, Post & Packing for U.K. deliveries (overseas add 15%)

Name: . . . .

Address: . . . .

STONECHIP ELECTRONICS
"more ways to make more of your computer"

DEALER ENQUIRIES WELCOME Delivery approx 14 days

For the best hardware, the best software.
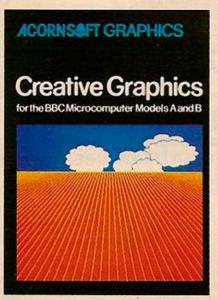
The BBC Microcomputer system is generally regarded to be the best micro in its price range you can lay your hands on. So, if you're thinking of buying one or already own one, you'll want to know about the software that's been specially designed for it.
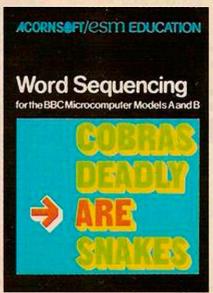
Not surprisingly, it's made by Acornsoft, the software division of Acorn Computers Ltd., who designed and built the BBC Microcomputer. So naturally you can expect the highest quality software with the built-in ingenuity to fully exploit the BBC Micro's potential.

Further education for everyone.

Creative Graphics, which includes the book 'Creative Graphics on the BBC Microcomputer' (price £17.45), provides 36 programs on cassette producing a spectacular range of pictures and patterns in full colour, including animated pictures, recursively-defined curves and three dimensional shapes.

Word Sequencing (price £11.90) contains three word sequencing programs on cassette. Each program presents a series of jumbled words which must be arranged on screen to form
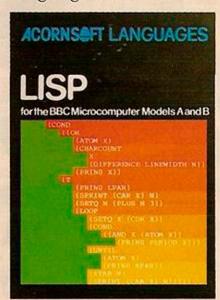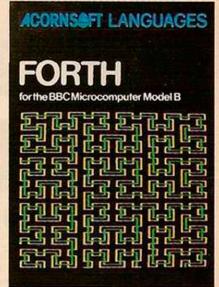
either a proverb, nursery rhyme title or a sensible sentence.

Learn more languages.

LISP (price £24.35) is the fundamental language of artificial intelligence research.

ACORNSOFT LANGUAGES
LISP
for the BBC Microcomputer Models A and B

ACORNSOFT LANGUAGES
FORTH
for the BBC Microcomputer Model B
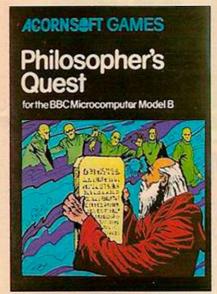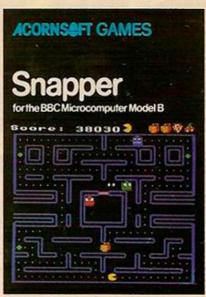
It consists of 5.5K of machine code interpreter, plus 3K of initialised LISP work-space containing utilities and constants. It comes complete with a book that introduces you to programming in LISP, as well as some fascinating applications.

FORTH (price £24.35) is a complete implementation of the FORTH language to the 1979 standard specification for the BBC Microcomputer Model B. This much acclaimed programming language is also accompanied by a specially written book explaining all you need to know.

Mind-boggling games.

Philosopher's Quest (price £9.95) is an advanced adventure in which you tell the computer what you want to do and it

ACORNSOFT GRAPHICS
Creative Graphics
for the BBC Microcomputer Models A and B

ACORNSOFT/esm EDUCATION
Word Sequencing
for the BBC Microcomputer Models A and B

COBRAS DEADLY ARE SNAKES

dk'tronics

ZX KEYBOARD FOR USE WITH 80/81 SPECTRUM

Our new cased keyboard has 52 keys, 12 of these are used for the numeric pad. The numeric pad offers some useful features, you can cursor with one hand and it will be a boon for anyone who enters a lot of numeric data. The pad is a repeat of the 1-9 keys plus it has a full stop and a shift key. The numeric pad keys are coloured in red, the normal keyboard keys are grey, with the case being black which makes the whole thing very attractive. The case measures 15 × 9 × 2½. The computer (either 80/81 or spectrum) fits neatly inside. You will have to remove the computer from its original case, it is then screwed to the base of the case. The case had all the bosses already fitted and the screw holes are marked. Also fitted inside the case is a mother board (81 model only) which allows 16K, 32K and 64K to be fitted in the case. All the connections are at the rear of the case i.e. Power, Mic, Ear, T.V. and the expansion port. The case is large enough for other add ons also to be fitted inside. One of these could be the power supply, then you could very quickly fit a mains switch, or a switch on the 9V line. This means you have a very smart self-contained unit. This case does not stop you from using any other add-ons that you may have e.g. Printer etc. We are convinced that this is the best keyboard available at present. It offers more keys and features than any other keyboard in its price range.

£45
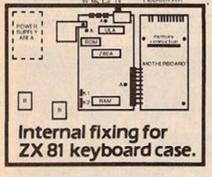
NOTE
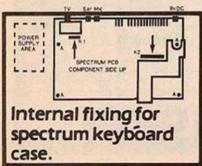
The case can be purchased separately with the keyboard aperture uncut, so if you have one of our early uncased keyboards, or in fact any other suppliers' keyboards, these could be fitted. The keyboard is connected to your computer by a ribbon cable and this has connectors fitted which simply push into the Sinclair connectors. It is a simple two minute job and requires no electronic skills. This keyboard does not need any soldering. Please specify on order whether you require the ZX 81 or Spectrum case.

SPECTRUM MODEL

This is supplied with spectrum legends, and a slightly different base for fitting the spectrum inside, again all the connectors are at the rear of the case and there is plenty of room for the power supply (and other add-ons). Should you want to change, we can supply both the Spectrum legends and details of updating your case which will enable modification from the ZX 81 to spectrum. PLEASE specify on your order whether you require the ZX 81 or spectrum case.

KEYBOARD (SPECTRUM/81)

Internal fixing for ZX 81 keyboard case.

Internal fixing for spectrum keyboard case.

SPECTRUM LIGHT PEN

The pen enables you to produce high resolution drawings on your own television set, saving a vast amount of time over using basic programming statements, such as Plot, Draw etc. You can erase, modify and save drawings and it comes complete with software program. The superb light pen is available from D.K. tronics.

£19.95

SPECTR

16K Memory Ex
The 16K uses 4116
Static Ram and oc
using the Static Ra
speed low power
the computer. A
which comes to y
where. Position in

64K Memory Ex
All the above info
advantage lies in t
56K of usable me
the use of other a
8192-65536. The
Spectrum Memo
Upgrade your Spe
it is simply slipped
are supplied, and
time. The fitting r
same as Sinclair's

ZX 80~81 Spectrum HARDWARE

SPECTRUM/81 TOOLKIT

This is the toolkit which won acclaim in the feature in the August 1982 issue (pages 29 and 30) of Sinclair User. "It is the most impressive program, fast in execution with clear and full instructions...it stands out from the rest of the field." The ZXED is a powerful editor for use on the expanded ZX81. It is intended for use by the serious BASIC programmer and offers several useful and time saving features most helpful during all stages of program development. The facilities provided are as follows: ALTER, BYTES, COPY, DELETE, FIND, HELP, INSERT, KEEP, MOVE, RENUMBER AND VERIFY. The Spectrum Toolkit contains most of the features above plus autoline numberer and append, and will run in the 16K and 48K spectrum.

Both at only £6.95

4K GRAPHICS ROM   £24.95

The DK Graphic module is our latest ZX 81 accessory. This module unlike most other accessories fits neatly inside your computer under the keyboard. The module comes ready built, fully tested and complete with a 4K graphic ROM. This will give you an unbelievable 448 extra pre-programmed graphics, your normal graphic set contains only 64. This means that you now have 512 graphics and with their inverse 1024. This now turns the 81 into a very powerful computer with a graphic set rarely found on larger more expensive machines. In the ROM are lower case letters, bombs, bullets, rockets, tanks, a complete set of invaders graphics and that only accounts for about 50 of them, there are still about 400 left (that may give you an idea as to the scope of the new ROM). However, the module does not finish there, it also has a spare holder on the board which will accept a further 4K of ROM/RAM. This holder can be fitted with a 1K/2K/RAM and can be used for user definable graphics so you can create your own custom character sets.

WHY WAIT?
ORDER TODAY FOR FAST DELIVERY

FLEXIBLE RIBBON CONNECTOR

If you have ever had whiteouts or system crashes this could be the answer. It stops the movement between the computer and the RAM expansion, it is supplied with a ribbon, 6 inches long, with a male connector at one end and a female at the other, at only

£10

16/64 MEMORY FOR ZX 81

16K Memory £22.95
16K (uncased) £19.95
64K Memory £52.95
64K (uncased) £49.95

The above illustration shows the casing for the 16K or massive 64K.

UM MEMORY

ansion £22.95

Dynamic Ram Chips. We use the dynamic as they are much denser than ...py less space. They are also much cheaper than the equivalent product ...n. The Ram is manufactured with high quality materials, and uses high ...ms. It is supplied ready-built and only needs to be plugged into the rear of ...he components are fitted into holders. This massive add-on memory ...u fully assembled and tested is the cheapest 16K memory available any- ...memory from 16384 to 32768. (Same as the Sinclair memory.)

16K (UNCASED) £19.95

ansion £52.95

...nation on the 16K also applies to the 64K Memory Expansion, but the ...e 64K giving nearly FOUR times the memory. This advanced model has ...ory. In addition, the block from 8K to 16K can be switched out to enable ...-ons. The graphics ROM is to be used in this area. Position in Memory: ...ck From 8192-16384 is switchable.

64K (UNCASED) £49.95

y Expansion £35.00

...um to 48K of user Ram. The Spectrum memory expansion is simple to fit, ...side the case, and then only requires plugging in. Full fitting instructions ...e only tool you will need is a screwdriver and just two minutes of your ...uires no electronic skills. Position in memory from 32768 to 65536. (The ...grade to 48K).

THE Computer Fair

Personal computers
Home computing
Small business systems

Sponsored by *Practical Computing* and *YOUR COMPUTER*

EARLS COURT
16-19 JUNE 1983

Opening times
Thursday: 1pm-6pm
Friday and Saturday: 10am-6pm
Sunday: 10am-5pm
Admission – Adults £3.00
Children (under 16) and Senior Citizens £2.00

See June issue for
£1.00 reduction voucher.

RETURNING BY POPULAR DEMAND BRITAIN'S BIGGEST MICRO SHOW

The Computer Fair gives you an unequalled opportunity to see and compare first-hand the enormous range of personal and home computers, small business systems, software programmes, cassettes and scores of computer games – decide for yourself how much or how little it takes to build up your own personal computer system.

Featuring... The Sinclair Village · Micro Mouse Contest · Club Avenue

---

Arrangements have been made with British Rail who offer a special ticket which includes entrance to the exhibition and return travel to a London B.R. terminal station and your county. All inclusive ticket prices are:–

SUPER SAVINGS

*Specially Reduced Rail Exhibition Fares

	Adult	Child		Adult	Child		Adult	Child
Avon	£14.50	£7.75	Hereford & Worcester	£13.00	£7.00	Shropshire	£17.00	£9.00
Bedfordshire	£7.40	£4.20	Hertfordshire	£5.20	£3.10	Somerset	£16.00	£8.50
Berkshire	£7.00	£4.00	Isle of Wight	£13.50	£7.25	Staffordshire	£15.50	£8.25
Buckinghamshire	£7.40	£4.20	Kent	£8.00	£4.50	Suffolk	£10.00	£5.50
Cambridgeshire	£8.50	£4.75	Leicestershire	£12.50	£6.75	Surrey	£6.00	£3.50
Cornwall	£28.00	£14.50	Lincolnshire	£16.50	£8.75	Sussex	£8.00	£4.50
Devon	£22.00	£11.50	Greater London	£5.00	£3.00	Warwickshire	£13.50	£7.25
Dorset	£14.50	£7.75	Norfolk	£14.50	£7.75	West Midlands	£14.00	£7.50
Essex	£5.80	£3.40	Northamptonshire	£9.00	£5.00	Wiltshire	£12.50	£6.75
Gloucestershire	£13.00	£7.00	Nottinghamshire	£14.50	£7.75	Greater Manchester	£21.00	£11.00
Hampshire	£10.00	£5.50	Oxfordshire	£8.50	£4.75			

Further details from: **The Travel Centre, Kings Cross Station, London NW19AP Tel: 01-278 2477**

Or fill in and return the coupon below.

HOW TO GET THERE

By Underground:– Earls Court Underground Station is adjacent to the Exhibition Centre. The station is on the District and Piccadilly lines.

By Bus (London Transport):– Northbound 31 and 74 buses pass the main Warwick Road entrance to the Centre. Southbound 30 and 74 buses pass the West Brompton entrance.

Car Park:– There is space for 1,300 cars in surrounding car parks.

British Rail Exhibition Tickets.

Please send

.... adult tickets at £ ....

.... child tickets at £ ....

The Computer Fair Personal computers Home computing Small business systems

From ....station.

Cheques made payable to British Rail.

Access, Barclaycard or Amex Card No. ....

Name ....

Address ....

Send to: **The Travel Centre Kings Cross Station London NW19AP**

Advance Party Bookings

For groups of 20 people or more

Adults: **£2.00**

Children: **£1.00**

(Plus one free ticket per 20 sold for the organiser or teacher).

Advance Party Bookings

.... adult tickets

.... child tickets plus one free ticket per 20 ordered.

Enclosed remittance of £ .... made payable to Reed Exhibitions.

Name ....

Address ....

Return to: **The Computer Fair Tickets, Reed Exhibitions, Surrey House, 1 Throwley Way, Sutton, Surrey SM1 4QQ.**

THE MEN WHO INVENTED ME WERE
CLEVER ENOUGH TO MAKE ME THINK
IN 'FORTH' (IT'S 10 TIMES FASTER
AND 4 TIMES MORE COMPACT THAN
'BASIC').

YET THEY'RE DUMB ENOUGH TO SELL
ME FOR £89.95!.

Richard Altwasser and Steven Vickers are the men who invented the Jupiter Ace.

After years of designing micro-computers that use BASIC (both men played a major role in creating the ZX Spectrum), they abandoned it in favour of FORTH.

FORTH is just as easy to learn as BASIC. Yet it's a faster, more compact and more structured language that educationalists and professional programmers alike prefer.

So the Jupiter Ace is the only micro-computer you can buy that is designed around FORTH.

Using it, there's little fear of accidentally 'crashing' programs halfway through and having to start all over again (a common fault with BASIC). The Jupiter Ace's comprehensive error checking sees to that.

The Jupiter Ace has a full-size keyboard, high resolution graphics, sound, floating point arithmetic, a fast, reliable cassette interface, 3K of RAM and a full 12 month warranty.

You get all that for £89.95. Plus a mains adaptor, all the leads needed to connect most cassette recorders and T.V.'s, a software catalogue (35 cassettes available, soon to be 50), the Jupiter Ace manual and a free demonstration cassette of 5 programs.

The Jupiter Ace manual is a complete introduction to personal computing and a simple-to-follow course in FORTH, from first principles to confident programming.

Plug-on 16K and 48K memory expansions are also available, at very competitive prices. (There'll be a plug-on printer interface available soon, too.)

It'll take you no time at all to realise how clever Richard and Steven were to design the Jupiter Ace around FORTH. And even less time to realise what a silly price £89.95 is to charge for it.

Technical Information

Hardware
Z80A; 8K ROM; 3K RAM.

Keyboard
40 moving keys; auto repeat; Caps Lock.

Screen
Memory mapped 32 col x 24 line flicker-free display upper and lower case ascii characters.

Graphics
High resolution 256 x 192 pixel user defined characters.

Sound
Internal loudspeaker may be programmed for entire audio spectrum.

Cassette
Programs and data in compact dictionary format may be saved, verified, loaded and merged. All tape files are named. Running at 1500 baud.

Expansion Port
Contains D.C. power rails and full Z80 Address, data and control signals. Can connect extra memory peripherals.

Editor
Allows complete editing and listing of compiled programs.

Please send cheque/postal order to:
Jupiter Cantab, 22 Foxhollow, Bar Hill, Cambridge CB3 8EP.

Jupiter ACE

YOUR LETTERS

SPECTRUM TV PROBLEM

Many Spectrum owners have not been able to obtain a colour display on certain colour televisions. They have a sharp monochrome display at one position of the TV tuner and distorted colour display at another. My solution is as follows: First, remove the five screws from underneath the casing and turn the computer upright; then carefully lift the top part of the casing so that it is just clear of the circuit board. Move it down over the rest of the casing taking care not to tension the keyboard ribbon cable.

Between the ULA and PAL encoder — see introductory manual page 29 — you will find four adjustable screws. These are used to tune the two separate outputs of the ULA which are related to the contents of the display file and the colour attributes. Next, connect the computer to the television as normal and create a colour display, for example

BORDER 1: PAPER 2: CLS

Note carefully the initial positions of the screws mentioned previously. Using a screwdriver adjust the top three of these screws until the colours appear on the display, starting with the resistor marked 2.2Kr. If after a period of time the colour disappears then further adjustment is needed. The bottom screw may also be adjusted if necessary. If this procedure fails then return the screws to their original positions. Finally, reassemble the casing, turn over and replace screws.

These alterations are both simple and perfectly safe and should put life into many a Spectrum display.

Mark Nolan, Malahide, Dublin.

MAZE CRAZE

A number of people seem to have had a problem with Maze published in the March edition of *Your Computer* The misunderstanding probably lies in line 5008, which reads

5008 LET d$ = "↑ <>v"

The string in this line should contain four characters — up-arrow, less-than, greater-than and lower-case v. The symbol in the middle is not the Sinclair unequal sign, although the two look the same on the listing.

In addition, it is worth typing Clear before saving the final version on tape — this prevents all the variables being dumped on to cassette.

Colin Carruthers, Edinburgh.

THE CURE

I built my ZX-81 from a Sinclair kit. After some teething troubles I had about six months of daily use without a hitch. Then the loading problems started. The loading was hit and miss, the trend became an increasing number of missed loadings. Something had to be done.

All the favourite cures had been tried, different volume settings; cleaner tape passed through; different types of tape; clean heads with meths and cotton wool buds; developed eye ache looking at loading patterns on the screen in an endeavour to interpret them; purchase and use of head demagnetiser.

It may sound silly but I suspect that many other computer "nut cases" are also guilty of not using the Tape Recorder for the purpose for which it is intended. During a

bout of frustration I listened to one of my favourite recordings and there was a lot of background hash. At first I accepted the poor signal/noise ratio as the quality standard of the recorder. But I became sure that the recorder did a better job some months ago. What if the computer was upset by this hash? I rooted out a capacitor substitution box and connected between the Ear input and earth of the recorder. The best de-coupling value was found by a marked reduction in the noise.

I obtained a suitable capacitor and soldered it into the recorder. That was over six months ago and loading has been no problem.

G Winterburn, Lytham, Lancashire.

SINCLAIR TRICKS

I have found out three things which may prove useful to Spectrum owners. The first is how to break into programs, which upon loading jump to some machine code. Type in the following.

999 Merge "" STOP

Load in the program, and when the loading is completed, the computer will display a Stop error code. You now have access to the program. The second, and most useful thing I found out, is how to disengage the Break key.

The trick in doing this, is to Poke down the screen to 24 lines deep at the beginning of your program. At any attempt to break into the program the screen will fill, and the computer will get lost in an endless loop. This happens because when the Break key is pressed, an error code appears. As the computer is used to printing on the bottom two lines, it will not like it if the two

lines are not accessible. This results in a crash.

If at any time in the program you do need to re-engage the Break key, include a line which Pokes back to two lines. The pokes are as follows:

POKE 23659,0 disengages
POKE 23659,2 re-engages

The third thing I found out is how to print on the bottom two lines. The function "Print # 1; "Hello"" will print hello on the bottom two lines. This is best demonstrated by typing in this short program.

10 Print # 1; CHR$ (RND*127+33): GOTO 10

Random characters could be printed on the bottom two lines. As the text goes over the line, the area expands to three lines, then to four, and so on. The machine code equivalent of this instruction is RST16, which prints the CHR$ of register A. I hope these few tricks will prove handy to Spectrum owners.

D J Whale, Harlow, Essex.

BBC MONITOR

Richard Harris's monitor program for the BBC Micro published in March *Your Computer* is excellent but happens to have one mistake. The label

650 .back DEC &80

The monitor can be further enhanced by printing the ASCII values when listing an eight-by-eight block and this is achieved by changing the following lines.

```
40 data1 = P$ + 965:data2 = P$ + 1120:
   data3 = P$ + 1275
50 mess = P$ + 1430:XYmess = P$ +
   1645:pmess = P$ + 1680
60 data4 = P$ + 1685:sptab = P$ + 1755
   :REM SET BASE ADDR OF DATA
```

These lines create the extra space required for new code inserted to print the ASCII values in block print. Following additions are required to achieve this. Replace lines 920 to 1110 with the following:

```
920 .block
930 LDA #8
940 STA &82
950 LDY #0
960 .cd LDX #8
970 STX &83
972 LDA &80
973 STA&84
974 LDA&81
975 STA &85
980 JSR newlin1
990 JSR addrpr
1000 .bc LDA (&80),Y
1010 JSR print
1020 LDY #1
1030 JSR space
1040 INC &80
1050 BNE ab
1060 INC &81
1070 .ab DEC &83
1080 BNE bc
1081 LDX #8:STX &83:LDY #2:
     JSR space
1085 .bf LDA (&84),Y:CMP # &20:
     BCC bw:CMP # &7E:BCC br
1086 .bw LDA # &2E
1090 .br JSR &FFEE
1091 INC &84
1092 BNE bh
1093 INC &85
1094 .bh DEC &83
1095 BNE bf
1096 DEC &82
1100 BNE cd
1110 JMP ff
```

Finally change lines 2430 to 2440 to:

```
2430 .newlin:JSR &FFE7
2440 .newlin1:JSR &FFE7:RTS
```

These changes will allow the user to locate data areas easily and will also help him to look more closely at the BBC OS and Basic Roms. I hope other users find these changes of use.

Deepak Laxman, Godalming, Surrey.

DEFENDING TIM

In response to A Jaques' letter — March edition — Mr. Hartnell was merely pointing out various ideas which the reader may put to good use.

Perhaps if Mr. Jaques had spent many hours of his valuable time writing programs he might understand what piracy can do to a company: his attitude typifies the person who simply wishes to have endless hours of fun without giving those who make it possible their just

NOSTALGIA AND PROGRESS

Thank you for your timely words of wisdom in the March issue editorial in respect of the current low-price computer war.

I own a UK-101, and over several years have upgraded it to 32K RAM, 16K ROM, a multi-dimension screen format including a useful 32 × 64, 8912 PSG, a PIA, VIA etc, with the inclusion of the Cegmon screen editor and machine-code monitor. I was beginning to develop an inferiority complex being restricted to block graphics only. As a matter of interest I ran the speed program in the Oric versus Spectrum article and was pleasantly surprised to achieve Load Array and Sort Array times of eight and 190 seconds respectively taking up 225 bytes of program space and 4041 bytes of variable space. The machine runs at 1Mhz which can also be upgraded.

Instead of buying one of the new plastic-case tortoises I will now return to my steel-clad monster — although not up to the size of the Manchester Mark I computer — and build a high-resolution and colour board with its own RAM leaving me with a full 32K memory less workspace — about 770 bytes — while I wait for the "ultimate" computer.

N L Smith, Stoke-on-Trent, Staffordshire.

reward. If Mr Jaques would look in the March 1983 Response Frame he would see there a letter written by someone who had tried to break into a program to copy it.

I regard *Your Computer* as the most user-friendly magazine available and quite simply the best at helping the user in trouble. Perhaps if Mr Jaques had experienced loading and output problems and suffered the frustration of it all then he would understand how valuable a service *Your Computer* is. I did not find any points technically related to a machine in Mr Jaques' letter.

Michael Gregory,
Guisborough,
Cleveland.

BELGIAN VIC

*L*iving in Belgium, I am the proud owner of a Vic-20. When attempting to load a program from cassette, it can be very helpful to listen to the cassette signals. This can be achieved by simply connecting a standard earphone to the Vic's cassette connector. It does not need an amplifier and cannot damage your Vic at all. Just connect the earphone leads parallel on pin A1 — ground — and pin D4 — read — of the cassette connector. See the user manual on page 151 for the pin configuration. Once connected you will both hear the loading and saving of your programs.

The synchronising header, checks, um, bytes and double storage are three good error-checking techniques used by Commodore to make their tape system very reliable.

Manuel Allaert,
Ostend,
Belgium.

FAST GRAPHICS

I apologise to readers using the machine-code routine described in my article on fast animated graphics for the Spectrum who have found that it does not work correctly when "Width" is greater than 16-bits. The culprit is a missing instruction LD (IX+9), 8 which should come between the instructions EX HL,DE and L4 DEC (IX+8) in listing 3. If you have your original version of the machine-code routine "Animate" on tape, then you should obtain a corrected version as follows: reset RAMtop to 65159 — 48K — or 32389 — 16K — with the appropriate clear command. Load your old version of "Animate".

Enter and run this Basic program:

```
10 Let A = 65160 (32390 for 16K)
20 POKE A + 158, 24
30 POKE A + 159,37
40 FOR B = A+197 TO A + 204
50 READ C : POKE B, C
60 NEXT B
70 DATA 78, 235, 221, 54, 9, 8, 24, 211
80 SAVE "ANIMATE" CODE A, 205
```

Robert Newman,
Oundle,
Peterborough.

YESTERDAY YOU WERE impressed by your bargain buy — £200 for a whole microcomputer. Today the smile of the proud new owner fades on your face as you see the same machine at the revised official price of £150. You rush to the calendar: it's too early for the summer sales, and April Fool's Day is long past. Then you recall something you read in last month's *Your Computer:* a Commodore spokesman, when asked to deny persistent rumours that new technology makes the £350 Commodore 64 cheaper to produce than the old £130 Vic-20, would only say that price "was all to do with what the market will bear — not to do with what it costs to make".

This is a clear statement of fact and nothing which should astonish or outrage us. After all British motorists have to pay more than the rest of Europe for the same cars.

Why then should we be surprised to learn that in the vast U.S. market, the ZX-81 — or Timex 1000 — can cost as little as £36 even though most Americans have larger disposable incomes, that the Stateside Spectrum is less than £100 for a machine with a higher specification, and that similar stories can be told about Ataris, TI-99/4s and any other micros you care to name?

You remember being told and not believing that the components inside a Spectrum could cost Sinclair as little as £15 and you begin to wonder if the £50 price cut after you bought your micro could just as easily have been £75 or perhaps even £100.

Before writing off micro pricing as a black art, you should understand how it works. Prices drop because products are in competition with newer or cheaper technology. If a micro-maker has the skill to be first on the market with a new kind of product, that company can inflate the asking price to as high a level as it believes the market will take. Then, as the competition arrives to share the pickings, the first company will deflate the price to try and increase the volume of sales and so maintain or even improve profits.

As the market grows, production costs tumble partly because the more integrated circuits the chip-builders produce, the smaller the percentage of faulty circuits — and that means increased yields and lower unit costs. Of course you can come a cropper if the original price is too high. The excellent TI-99/4 was the first 16-bit colour home micro and cost about £750 when it was introduced in 1980. As other machines appeared, the TI-99/4's price had to be dropped progressively to its current level of £150 to remain competitive with the newcomers. Some would argue that its original price asked too much of the U.K. market and damaged the micro's prospects in the long run. A £350 price tag for a machine of such a calibre in 1980 might have given the TI-99/4 a vast user and software base in Britain.

Clearly there are concealed costs beyond the price of components. Companies have to allow for research and development, labour, production overheads, packaging, distribution, marketing, margins for retailers and even the expense of maintaining the jammed switchboard of the customer complaints department. But when you add all that up, the gap between costs and price can still be very large. Perhaps that explains why so many new companies are bravely rushing into computing to fill the breach.

How to write for *Your Computer*

We called this magazine *Your Computer* precisely because we welcome your views, tips and hints and even your criticism of machines and software in general. If you would like to see your name in print, whether on a Software File program or a full-blooded article, here is how to go about it. Ideally, all articles should be typed double-spaced on one side only of uniform sheets of paper. If listings can be dumped directly from a printer — you can always use a friend's or user group's — this minimises the risk of error. In a perfect world a cassette would accompany the article. That considerably speeds up the checking process. Not only do you get to air your own discoveries and opinions, but we will even pay you for the privilege. We pay £35 per published page — that's as it appears in the magazine and includes illustrations.

Why not give it a try? You have nothing to lose but your postage.

Premier disc system — a first for Dragon 32

NOW THERE IS a disc system for the Dragon 32. Premier Publications has incorporated a Disc Operating System and disc controller in a single cartridge which plugs into the games cartridge port.

The Disc Operating System supplies a comprehensive set of commands which appear to the user as an extension of Basic. As well as standard facilities for creating and deleting files, reading the directory, disc formatting, renaming files, there is also provision for opening up to eight random and sequential files simultaneously.

Two unusual features are the Find and Config commands. Find enables data files to be searched for a string match. Config allows the user to configure the system to run with almost any type of floppy-disc drive available. The number of tracks, sides and sectors per track on a disc can all be specified to allow the system to support 3, 5.25 and 8 in. drives. In addition the controller cartridge can cater for up to four drives at a time, in any combination.

The cartridge comes with a 74 page manual and costs £99.95. Premier Publications is also selling complete systems, starting with the Delta 1 at £299.95.

This includes the controller, cables, and a 40-track Canon disc drive with a capacity of 100K.

Premier Publications is at 208, Croyden Road, Anerley, London SE20 7YX.

Millionth U.K. micro sold

THE END OF APRIL marks a milestone for home computing — with the sale of the home computer in Britain. There are more ZX-81s out there than anything else followed by Spectrums, Vic-20s, BBC Micros, Dragons, TI-99/4s and a dozen other computers.

BBC Music Processor

QUICKSILVA MUSIC Processor turns your BBC Micro into a tape recorder and synthesiser for just £14.95. Andy Williams the author of the program, started working on it in May last year. He says that if he had worked full time it would still have taken him six months to complete.

Business Press

THE NAME of *Your Computer's* parent company has been changed from IPC Business Press Ltd to Business Press International Ltd. This change has been made to reflect the very wide range of markets covered by the 100 publications of the company, and to identify its prime position as the world leader of business publishing.

Spectrum goes organic while ZX-81 grows to one megabyte

THIS MONTH Basicare is releasing an interface which will enable Spectrum owners to use most of the modules in the company's Organic Micro system which is now available for the ZX-81.

These include a CMOS RAM module backed by battery for non-volatile program storage, Toolkit, an interface to take up a 8K of utility programs stored in EPROM or ROM, and a sound synthesiser module. There is also a general purpose interface providing 24 input/output lines and a Centronics interface to link the ZX-81 or Spectrum to most printers.

On the ZX-81 the printer

interface is accompanied by a program which prints ZX-81 graphic characters on Epson and Seikosha printers.

The modules can be stacked on top of each other and are interfaced to the computer through the Persona module which communicates with the other devices via a 64-way bus.

One feature that sadly will not be available for the Spectrum is the Minimap module. This enables ZX-81 owners to access up to one megabyte of RAM. Only 64K can be read at one time but up to 16 pages of 64K can be switched in and out using a single Poke instruction.

Analogue to digital, and digital to analogue converters for both the ZX-81 and Spectrum will also be released. Future developments include a joystick controller and a fast cassette loading device.

The Personal module costs £30.25 and the Minimap £35.95. Prices for the other devices range from £22.20 for Toolkit to £76.25 for 64K RAM. Basicare is at 12 Rickett Street, London SW6. Tel: 385 2135.

Spectrum owners can now link their micro up to the £149 Tandy colour printer. Softest is selling an interface package for £35. It includes the necessary hardware together with cables and three programs which allow user to access the Tandy printer commands. A high-level driver program to make graph plotting easier is under development and will be available for £5.00. Softest is at 10 Richmond Lane, Romsey, Hampshire, SO5 8LA.

Ikon's Hobbit for the BBC Micro runs rings around conventional cassette storage

YOU'VE READ THE BOOK, you've seen the film, now buy the floppy tape drive. Hobbit is the name of a new storage unit for the BBC Micro which is faster than cassette and cheaper than disc storage. It reads and writes at almost seven times the rate of the standard cassette, and also offers many of the housekeeping facilities of a disc system.

Ikon Computer Products, which makes the Hobbit is supplying it with an EPROM which contains the floppy tape equivalent of a disc-operating system and takes care of such tasks as locating a program on tape and updating a directory. All of the standard BBC file commands are supported as well as a few extra commands which are specific to the Hobbit.

The Hobbit has similarities to Sinclair's Microdrive but whereas the Microdrive will use a continuous loop of tape, tape on the Hobbit can be moved backwards and forwards. The average access time is 20 seconds with a maximum of 90 seconds.

Two drives can be connected to the BBC through the user port. Each takes a cassette with a capacity of 60K per side.

Perhaps Ikon should have considered calling the £155 Hobbit after another furry animal — the rabbit in *Magic Roundabout* — because Ikon is based at Kiln Lake, Laugharne, Dyfed — which used to be the home of Dyan Thomas.

80K for £300 coming soon
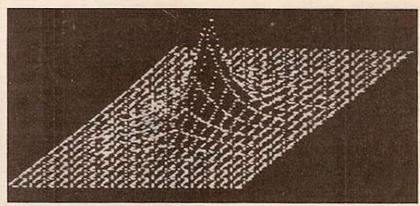
LATE NEWS of an 80K home micro about to be announced by one of the world's biggest micro makers. The machine will set new standards in BBC Micro, Commodore 64 country and will cost around £300. More details next month.

High resolution Sinclair tape from self-taught schoolboy

RICHARD TAYLOR, a 14-year-old schoolboy, has found a way to improve the ZX-81's 64×44 pixel resolution without using any extra hardware. Until now high resolution on the ZX-81 could only be obtained from high-res graphics boards costing over £50. Richard Taylor's program, High Resolution, comes on a cassette and gives a resolution of 256 by 192.

The program takes up around three-quarters of a K and creates its own display file which consumes a further 6K. It allows you to plot points, invert points, save and load a high-res picture, and also to define new characters.

Like most of the other programming whizzkids Richard Taylor is entirely self-taught. He finds rewriting the operating system easier than devising games. "There is no point in doing what everyone else is doing", he explained and added "I like to make a machine do what it's not designed for". He has now moved on to the Spectrum and is working on a program to give the Spectrum a genuine 36 colours.
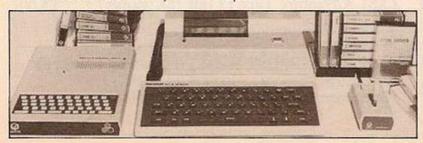
The High Resolution program costs £5.95 and is available from Computer Rentals, 140 Whitechapel Road, London E1.

Android Attack on the trail of Dragon Donkey King

FLUSHED WITH the success of its top-selling Dragon program Donkey King, Microdeal has now released five new programs for the Dragon. Android Attack is based on the arcade game Beserk, includes some limited speech synthesis and routinely announces intruder alert or charges the player with cowardice. Galactic Ambush and Cosmic Zap are Microdeal's versions of Galaxians and Space Fortress. Program packs 4 and 5 each contain nine short programs. These tapes are available from Microdeal and Dragon dealers for £8 each.

Remember the MPF II — the £269 colour computer we scooped last October? At the time our reviewer complained that it was a good machine let down by a dreadful keyboard. Here it is with the £36 full-size keyboard, joystick, £185 thermal printer and some software all of which are now available all over the country from the Spectrum dealer chain.

LOAD AND GO WITH YOUR DRAGON

32 PROGRAMS FOR YOUR DRAGON 32

* ★ Games Pairs
    Pontoon
    Snakes and Ladders
    Ski Run
    Castle Walls
* ★ Homework helpers
    Sums Tester
    Vocabulary tester
* ★ Graphics section
    Full explanation of all Graphics
    commands with worked examples
* ★ Hints and Tips
* ★ How to debug programs
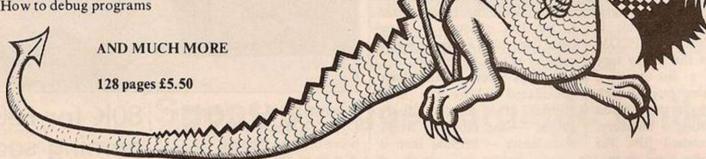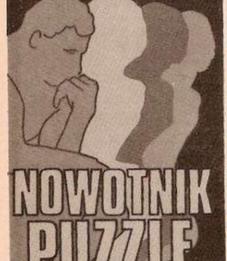
AND MUCH MORE

128 pages £5.50

PHIPPS ASSOCIATES

Dept A FREEPOST EM463 (No stamp required)
99 East St, Epsom, Surrey KT17 1BR.
Telephone 03727-21215. 24hr phone service.

Prices include postage but for air mail delivery in Europe add 90p (outside Europe add £1.70) per item.
Access and Barclaycard accepted

NOWOTNIK PUZZLE

For either the 16K Spectrum or 16K ZX81, this is an original challenge for anyone. The screen becomes the playing board for this addictive puzzle. *Machine coded* for smooth and instant responses, it offers four levels of difficulty. The computer breaks and shuffles a two-by-two coloured square - have you the skill and mental agility to restore it? *"Very addictive"* – ZX Computing Feb/March 1983.

ZX SPECTRUM

Knight's Quest (48K)	£5.95
Nowotnik Puzzle (16K)	£4.95

KNIGHT'S QUEST

A full-blooded adventure for the 48K Spectrum using split-screen *graphic pictures* and a scrolling text window. You must find Merlin's lost treasure, battling elves, scorpions, dragons; rescue a princess from the evil Wizard of Trill – it goes on and on! Over 120 locations plus a full English command line scanner, *machine coded* for fast recognition. Truly state-of-the-art! Also available for the 16K ZX81.

ZX81

Knight' Quest (16K)	£4.95
Nowotnik Puzzle & Other Diversions (16K)	£5.00
ZX81 Adventure Tape (3 full 16K games)	£5.00

PHIPPS ASSOCIATES

Dept A FREEPOST EM463 (No stamp required)
99 East St, Epsom, Surrey KT17 1BR.
Telephone 03727-21215. 24hr phone service.

Prices include postage (outside Europe add £1.00) per item.

Acornsoft Castle of Riddles: the knights errant receive rewards

A DAWN CROSS-COUNTRY DASH from Littlehampton on the Sussex coast to *Your Computer's* offices on the outskirts of London just failed to clinch first prize in the £3,000 Castle of Riddles competition for 34-year-old company director Colin Bignell. Only 20 minutes before Colin Bignell's car drew up outside the *Your Computer* headquarters, Peter Voke of Aylesbury, Buckinghamshire was safely delivering the winning entry into the hands of editor Toby Wolpe.

Third past the post was Peter Colbert, a Cardiff-based British Telecom programmer. He put his faith in his colleagues at the Royal Mail and entrusted his entry to the vagaries of first-class. He had solved the riddles in six hours — two hours faster than the other prize-winners. Both he and Colin Bignell each receive £400 of Acorn equipment, while Peter Voke won the £700 silver "King of the Ring" trophy and £1,500 of Acorn hardware.

At the top of his shopping list is a dry ink-jet printer, followed by a small disc drive and a

From left: runner-up C P Bignell; *Your Computer* editor Toby Wolpe; winner Peter Voke; Acornsoft's David Johnson-Davies; and in third place P K Colbert.

second processor. The two runners-up have set their sights on disc drives, although Peter Colbert also finds the voice-synthesis chip a very attractive proposition. His six-hour session that solved the Castle of Riddles started on a Saturday morning

and ended early in the aftrnoon of the same day.

Although the cassette had arrived two days before, his wife was against his spending all night at the keyboard.

Away from the cut and thrust of the world of adventures,

"King of the Ring" Peter Voke confides that time spent away from the keyboard and, above all, moderation are his keys to success. He recommends sleeping sensibly — would-be champions should note that throughout the time he was wrestling with the Castle of Riddles he made sure he got his 10 hours a night.

Computer Club is here to encourage you to start your own local computer club or, if one already exists, to join it and become involved. We would like to hear of anything which has made your club a success, or of any projects or programs you are developing.

BIRMINGHAM CALLING

Birmingham is not only the venue for our Midland Computer Fair at the end of April — it is also the home of a computer club which can be heard broadcasting on local radio. Simon Beesley tunes in to West Midlands Computer Group.

FOR A SMALL club the West Midlands Computer Group is remarkably productive. It puts out a monthly magazine, Databus, has its own 15-minute slot once a week on Radio West Midlands, and also produces its own software. Databus packs into 10 pages a useful mixture of news, articles, programs, programming and software reviews — all contributed by club members.

The driving force behind the club is David Thomson who founded it last August. He teaches at a local school and was initially concerned to provide a forum for the computing interests of pupils once they had left school. He was assisted by three of his ex-pupils in setting up the club.

The three in question are all talented programmers and are considering marketing their software. With the present boom in commercial software sales no doubt it will not be long before they are rubbing shoulders with the Bug-Bytes and Quicksilvas of the business. Your Computer — ever quick to spot new talent — can boast that we gave one of them, David Griffin, his first break when we published his game Maze Munch in our December issue. This program for the BBC Micro was one of our most popular and as an amateur programmer myself I was rather disconcerted to learn that it had taken David Griffin just one hour to write.

From its connection with David Thomson's school the club has now broadened out and, according to Databus, includes no less than three astronomers among its 36 members. One of the astronomers uses a ZX-81 to control the movement of his telescope.

Most of the machines in evidence were BBC microcomputers. Holding its own with the BBCs was a customised ZX-81 in full concourse trim, complete with an add-on keyboard, a wooden casing, a Qsave tape-loader and Atari joysticks.

David Thomson suggested that one of the reasons for the BBC Micro's popularity among computer club enthusiasts was the extensive facilities it offers for communicating with the outside world. His own interests are concentrated, as he put it, on the back of the computer and he writes a regular series in the club's magazine on control applications.

At this meeting he demonstrated how to write to the user port on the BBC. Another member had brought along a board fitted with LEDs which plugged into the user port and showed which control lines were in operation.

Elsewhere in the room David Berry, the club's ZX-81 expert, turned his hand to showing a beginner how to write the software which would read joystick inputs on the BBC.

Another group was taking an unconventional approach to adventure games. One of the members had cracked the code which Acornsoft adventure programs use to store their phrases and sentences. He had with him printouts of all the responses that the computer could give in the Castle of Riddles and Philosopher's Quest — a philosopher's crib sheet.

As can be seen, within an informal framework, the West Midland Computer Group's activities are varied and enterprising. The club meets once a month in the Victorian splendour of a room at Birmingham's Central Hall. For more details you can contact David Thomson on Bromsgrove 32242.

Members of all ages cracking an adventure game.

Local society news

Basildon buffs
THERE is now a microcomputer club for the Basildon area. It meets fortnightly on a Tuesday evening at Healey Management Services, The Hemmels, Laindon, Essex. You can telephone Roger Sims on Wickford 63032 after 6.30pm.

Thurnscoe micros
AROUND 70 people attended the first meeting of the Thurnscoe and District Microclub in March. Not all were owners but a straw poll revealed interests in a wide range of machines, from the Spectrum and the Dragon to the Pet and Nascom III. Meetings are held every Wednesday evening at Thurnscoe Comprehensive School, Clayton Lane, Thurnscoe, South Yorkshire. For further information contact Peter Davis on 0709-893880.

Liverpool users
SINCLAIR owners in Liverpool can attend meetings of the ZX Users Club every Wednesday evening at the ZX Computer Centre, 17 Sweeting Street, off Castle Street, Liverpool 2. Anyone is welcome whether they have software or hardware problems, an interest in programs and games, or just a general interest. More details can be obtained from Keith Archer on 051-260-4950.

Enfield brigade
ENFIELD Computer Club caters for the novice and expert alike. Meetings are held on the first and third Monday of each month at Brigadier Youth Centre, Brigadier Hill, Enfield, Middlesex. The annual membership fee is £2 and a further 25 pence is charged at each meeting.

FIRST BYTES

'What does the term memory-mapped mean? I have seen it used to describe screen displays on computers.'

QUESTIONS
What is a memory map?

YOU CAN THINK of the computer's memory as being composed of a series of boxes. Each box is a memory location and can hold a number between 0 and 255. One area of memory — usually either 8K or 16K — is set aside for the computer's read-only memory" — ROM — which contains the system that allows the computer to interact with you, do calculations, put things on the TV screen, and so on. This area can only be read from, so it is rather like a jar with a lid on because you can see what is inside, by Peeking, but you cannot put new information in — that is called Poking.

One further area is reserved for the information which is displayed on the screen, and it is this area that you refer to in your question. The locations in this case are in "random access memory" — RAM — and this means that information can be read from and written to them. So you can both put new information in these locations — which amounts to putting new information on the screen — and see what is there. The computer's electronics translate the numbers in these locations into what will appear on the TV screen.

Now, there are two ways in which this can be done. If you have a ZX-81, for instance, you will find that the screen RAM — often called the Display File — contains the Codes of the characters which are to be put on the screen. It does not hold information about each and every dot — or pixel — on the screen.

This may be called a character-mapped screen. Before printing a character the computer needs to consult a section of ROM known as the character generator, which stores the dot patterns for each character.

On the other hand, owners of machines such as the Spectrum, Dragon or BBC Micros will find that the Display File holds information about the rows in each character — the bytes on the screen, where there are eight bytes to a character and every byte is eight dots — or pixels — wide. This is sometimes referred to as a bit-mapped screen since each pixel on the screen is assigned to a bit in memory.

If your computer's screen is truly memory-mapped this will mean that there is a set, unmoveable area of memory put aside by the micro. In the case of a character-mapped screen about 700 locations are reserved, and for a bit- or pixel-mapped screen the area will be at least eight times as large.

The number stored in the first location in the screen RAM usually determines what will appear in the top left-hand corner of the screen, the second location relates to the second position on the screen, and so on. In some cases the screen may be somewhat more complicated than this — as owners of the Spectrum may have discovered.

Sometimes, however, a computer will not set aside a fixed area but will instead keep enough locations free somewhere within its RAM — with a pointer to the start of this area being kept in a known place. In othr cases, and Camputers' Lynx is one example, the screen's memory may be quite separate from the RAM which is available to the programmer. In such cases the Display File may be in separate memory chips which cannot easily be read from or written to.

Lastly, some micros put aside a varying amount of memory for the screen depending on whether a text or graphics display is chosen. This may amount to your having the choice between a memory-frugal Text mode and a memory-consuming Graphics mode — the Oric 1 and Dragon 32 are just two examples here.

Tim Langdell

BEATING
PUT A REM

ONE OF THE advantages of writing your own programs, as opposed to typing other people's in, is that at least you know what the machine is meant to be doing — so debugging certainly should be easier. But it is amazing how quickly you can forget what a particular bit of code is trying to achieve.

One of the most powerful weapons in the debugger's armory turns out to be the lowly Rem statement.

This is very useful if it merely tells you what

I could do that...

KEYING IN

ALL VERSIONS of Basic allow you to enter information from the keyboard with Input. When Input is reached the program waits for you to enter something before continuing. It allows you to enter several characters at the same time and edit them before Input, but has the disadvantage that you must always follow your last character with Enter, which is slow and can be a nuisance when you only need to use a single key in both games or serious programs.

The alternative way of getting information from the keyboard, which is found in many Basic dialects, is the Inkey$ function. This can continually scan the keyboard and tell you instantly when a key is pressed. Inkey$ is also particularly useful in high-resolution graphics modes as it does not send you striaght back to the normal text screen as Input does. As the program does not automatically stop and wait for you to do something a loop must be made

around Inkey$ which is only left when a key is pressed:

```
10 A$ = INKEY$
20 IF A$ = "" THEN 10
```

Note that Inkey$ always gives you a String variable, and that the sign for an empty string is double quotes and a full stop.

If you want to enter a number and use it for calculations then you must convert the string into a numeric variable with the Val function:

```
30 A = VAL(A$)
```

If you want to use cursor keys to continuously control left and right movement you can convert the Inkey$ string to its ASCII value — 8 for left and 9 for right — and then simply add on the left and right movements to the current position:

```
30 A = ASC(A$)
```
(or A = CODE(A$) on Sinclair)
```
40 POSITION = POSITION + ((A=8)-(A=9))
```

Keith and Steven Brain

Relational and logical operators may sound awesome, but a computer would be lost without them. Here we explain them as simply as possible.

ALL ABOUT OPERATORS

Lynx winner, Dave Bull giving his two-year-old son early keyboard experience. Competition details on page 183.

A COMPUTER CAN only tell the difference between off and on. Sitting on the fence is not possible. It is therefore totally logical in the way it makes decisions, although of course it can only decide between options which you have decided to include in your program. All decisions are based on testing whether a particular condition — or test — is True or False. Many decisions can be made using only the simple relational operators:

=	equal to
>	greater than
<	less than
<>	not equal to
>=	greater than or equal to
<=	less than or equal to

Both numbers and strings can be compared, but with strings you must make sure that there are no invisible spaces hiding at the end of words. If you test whether

"FIRST " = "FIRST"

then the answer will be no, and similarly "BYTES" is not the same as "bytes". The last two relations are not absolutely necessary, particularly when dealing with numbers, as it is often as easy to write a program which does not need them.

```
10 IF A >= B .......
```

is equivalent to:

```
10 IF A > (B–1) .......
```

Comparisons of conditions may be made more exhaustive — not to say exhausting — by including the logical operators And, Or and Not which combine several tests together in a single program line but can become rather mind-bending.

The first logical operator, And, should be easy enough to understand,

IF test 1 AND test 2 TRUE then whole test TRUE

If both test 1 and test 2 are true then whole test is true but if either test 1 or test 2 is false the whole test is false:

```
10 IF TEMP = 0 AND DAY = 7 THEN
   STAY IN BED
```

and Or is not too bad either,

IF test 1 OR test 2 TRUE then whole test TRUE

that is, if either test 1 or test 2 is true the whole test is true:

```
10 IF MONEY = 0 OR SHOP = SHUT
   THEN HARD LUCK
```

but Not often causes chaos as it looks at first sight as if it works backwards.

IF NOT test 1 FALSE then whole test TRUE

If test 1 fails then the whole test is True: although the sense of the logic can be seen if you consider:

```
10 IF NOT (DAY=PAYDAY) THEN NO
MONEY
```

Keith and Steven Brain

THE BUGS
A STATEMENT IN IT

each bit of the program is doing. For instance:

REM PLOT VALUE

or

REM ASK NEXT QUESTION

Many errors show up immediately you read through the program, because what the Rem say and what the code is obviously doing are quite clearly different.

Rems are also very useful for keeping track of what you are using your variables for. It is a good idea to set aside the first 100 line numbers of each program for Rems, mainly to describe variables. Each time you add a new variable, go back and write a Rem saying what it is for:

40 REM H IS HEAD POSITION
50 REM T IS TAIL POSITION

A good way of developing your own bug-free programs is to keep running a program as you type it in. Apart from making the whole process more interesting, most errors are thrown up very soon after you have typed in the lines causing the problem, so it is easier to sort things out.

Another way of developing programs which makes tracking down errors easier is to make use of "stubs". This approach lets you sort out the main point of the program first, and the fiddly bits are sorted out later. A stub is a subroutine which does the fiddly bit in a provisional, simple way.

For instance, if each time an object hits the side of the screen you want to make an elaborate noise, you can put in a Rem

REM GO AND MAKE ELABORATE NOISE

at the appropriate point in your program. Under the remark you put a Gosub instruction off to a high line number, and at that line number you put a very simple routine which merely makes a feeble beep and then Returns.

When you run the program you will know that the noise is occurring in the right place, and you can give your full attention to sorting out the rest of the program.

It makes sense to put all your subroutines together, separately from the rest of the program flow, perhaps at the end of the program, with very high line numbers, say over 10,000. With some machines it is better to put them at the beginning, especially if the subroutines are used often, because the Basic interpreter may take longer to find line numbers towards the end of the program.

With subroutines it is especially important to include plenty of Rems, as you may be calling a particular routine from several different points in your program.

If you are adding lines to a program to help you write it which you intend taking out later — either Remarks, or perhaps Print statements — it is important to be able to get them out without introducing new errors. A very common error message is

UNDEFINED STATEMENT IN LINE n

which is caused by branching to a non-existent line. Sinclair Basic is unusual in that it lets you get away with this, assuming you mean the next highest line number.

To help me keep track of lines I might want to get rid of, I give them odd line numbers. Usually I use only even line numbers, and go up in increments of ten. So if I see a line 51 in my listing I am automatically suspicious about it once I have finished testing a program.

With Rems it is also easy to put recognisable characters in the line, for instance a string of asterisks, so you can find the lines you want to pull out easily later. Remarks do not have to be on a line of their own, and can be used after a colon to flag other testing lines. *Ian Stobie*

TELEWRITER™
the DRAGON 32 Word Processor

TELEWRITER

Telewriter is the powerful word processor designed specifically for the DRAGON 32 Computer. It can handle almost any serious writing job and is extremely easy to use. It has all the advanced features you need to create, edit, store, format and print any kind of text. With Telewriter you can quickly produce perfect, finished copy for letters, reports, term papers, articles, technical documentation, stories, novels, screenplays, newsletters. It is also a flexible and efficient way to take notes or organize ideas and plans.

51 × 24 DISPLAY

The DRAGON 32 is an incredibly powerful and versatile computer, but for text editing it has some major drawbacks. The small 32 character by 16 line screen format shows you too little of the text and, combined with its lack of lower case letters, bears little resemblance to the way text really looks on the page. Reverse video in place of lower case just adds confusion.

Telewriter eliminates these shortcomings with **no hardware modifications required.** By using software alone, Telewriter creates a new character set that has **real lower case letters,** and puts 24 lines of 51 characters on the screen. That's more on-screen characters than Apple II, Atari or TRS-80 Model III. That's more than double the DRAGON 32's standard display.

FULL SCREEN EDITOR

The Telewriter editor is designed for maximum ease of use. The commands are single key (or single key plus control key), fast, and easy to remember. There is no need to switch between insert modes and delete modes and cursor movement modes. You simply type. What you type is inserted into the text at the cursor, on the screen. What you see on the screen is always the current state of your text. You can move quickly through the text with one key cursor movement in all 4 directions, or press the shift key simultaneously for fast, auto-repeat. You can jump to the top or bottom of the text, and beginning or end of a line, move forward or backward a page at a time, or scroll quickly up or down. When you type past the end of the line, the wordwrap feature moves you cleanly to the next.

You can copy, move or delete any size block of text, search repeatedly for any pattern of characters, then instantly delete it or replace it with another. Telewriter gives you a tab

. . . truly a state of the art word processor . . . outstanding in every respect
— The RAINBOW, Jan. 1982

The only one with all these features for your DRAGON 32

51 column × 24 line screen display
Sophisticated full-screen editor
Real lower case characters
Powerful text formatter
Works with any printer
Special MX-80 driver
Requires absolutely
no hardware modifications
★ Tandy colour version
also available

key, tells you how much space you have left in memory, and warns you when the buffer is full.

FORMAT FEATURES

When it comes time to print out the finished manuscript, Telewriter lets you specify: left, right, top, and bottom margins, line spacing and lines per page. These parameters can be set before printing or they can be dynamically modified during printing with simple format codes in text.

Telewriter will automatically number A4 pages (if you want) and centre lines. It can chain print any number of text files from cassette without user intervention. You can tell it to start a new page anywhere in the text or pause at the bottom of the page.

You can print all or any part of the text buffer, abort the printing at any point, and there is a "Typewriter" feature which allows you to type straight to your printer. Because Telewriter lets you output numeric control codes directly (either from the menu or during printing), it works with any printer (Tandy, Seikosha, MX-80, Okidata, NEC 8023, C. Itoh 8510, Centronics, GE Terminet, Smith Corona TP-1, etc.). There's even a special driver for the Epson MX-80 that lets you simply select any of its 12 fonts and do underlining with a single underline character.

CASSETTE INPUT/OUTPUT

Because the Telewriter makes using cassette almost painless, you can still have a powerful word processor without the major additional cost of a disk. The advanced cassette handler will search in the forward direction till it finds the first valid file, so there's no need to keep retyping a load command when you are lost in your tape. The Verify command checks your cassette save to make sure they're good. You can save all or any part of the text buffer to cassette and you can append pre-existing files from those you have in the buffer already.

ASCII COMPATIBLE

Telewriter turns your DRAGON 32 into the most powerful, lowest cost, word processor in the world today. But that's not all. The simple ASCII conversion program provided with Telewriter means you can use the full power of the Telewriter editor for creating and editing BASIC and assembly language programs. It means you can use Telewriter to prepare or edit text files used with any data communications program.

Telewriter costs £49.95 on cassette and is

AVAILABLE FROM DRAGON 32 DEALERS NATIONWIDE
OR DIRECT FROM
MICRODEAL
41 TRURO ROAD, ST. AUSTELL, CORNWALL PL25 5JE
24 HOUR ORDERLINE 0726 67676

WE WELCOME ACCESS   AMERICAN EXPRESS
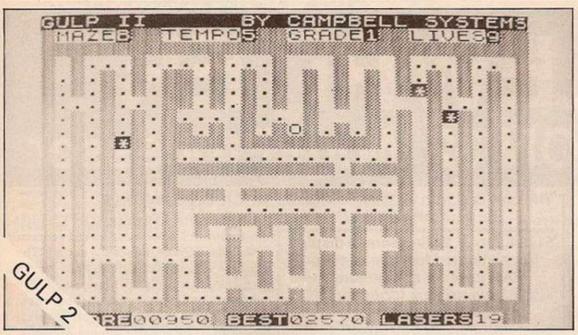
. . . one of the best programs for the Color Computer I have seen . . .
— Color Computer News, Jan. 1982

Apple II is a trademark of Apple Computer, Inc.: Atari is a trademark of Atari, Inc.: TRS-80 is a trademark of Tandy Corp.: MX-80 is a trademark of Epson America, Inc. DRAGON 32 is a trademark of Dragon Data Ltd.

GULP 2

SURVEY

All systems are go as
Peter Connor explores the
weird and wonderful
lengths to which software
houses go to create
programs for the ZX-81.

IT IS NOT easy to review software for the ZX-81 without using the weary old disclaimer "Given the limitations of the machine . . .", and I am not going to try. These limitations, as every owner must be well aware, are the primitive graphics and the absence of sound or colour. The limitations, of course, are far outweighed by the advantages; at about £50 the ZX-81 made home computing available to practically anyone who wanted it. After all, £50 is only 50 packets of cigarettes, or for younger readers about 350 Mars Bars.

To produce a version of Scramble, then, for the ZX-81 might seem as hopeless as trying to paint the Forth Bridge with a toothbrush. Yet it works surprisingly well. In Mikro-Gen's version all the usual features are there: the rugged terrain, guns, missiles and smart bombs, the different defence zones. If your gun seems to fire more like a peashooter than a Mig-21 you must remember that, given the limitations of the machine, this game is pretty good value.

Another old favourite produced for the ZX-81 is Invaders. Silversoft's and Bug-Byte's versions are, understandably, very similar. Both have different skill levels and the rather chunky aliens come fast and furious at the highest level. At the lowest level the games have a strange slow-motion effect, as if the invaders were on tranquillisers.

One thing games on the ZX-81 bring home to you is the magical and often hostile properties of letters. In Mikro-Gen's Tempest, a clever variant of Scramble, you have to survive as long as possible against the alien Spinners, Liners, Walkers and Asteroids. These aliens are none other than wandering letters and symbols; it is rather like being under attack from a psychotic typewriter keyboard. The Walkers — Ws — wander all over the screen while the Asteroids — formerly friendly asterisks — come straight at you. More dangerous than either of these are the Untouchables, one of whom bears a remarkable resemblance to the space bar. If you do not use your smart bombs, they will crush you. The game is in five sectors, through which the aliens increase in speed. The pleasing thing about Tempest is that Mikro-Gen have made necessity the mother of invention and used the ZX-81's simplicity to create a game of almost surreal charm.

Galaxians from Quicksilva has something of

the same interest. Here, the first wave of swooping aliens consists of flying Vs. But if you think those are dangerous, you'd better look out for those Ws lurking at the top of the screen. W is a genuinely nasty letter. Given the limitations et cetera, et cetera, Galaxians is a good version of the standard game. It is fast and if you miss the sound effects you could always make them yourself.

Several other arcade favourites are now available for the ZX-81. DJL Software has produced Frogger and Psion has the same game as Frogs. Both are good versions, with the logs really rolling at the highest levels. DJL's version, though, has the slight edge; as the game progresses you encounter the additional hazards of snakes and alligators. There is also the extra task of rescuing cute baby frogs from the logs on your way across. It is heartening to see software houses encouraging a sense of civil responsibility.
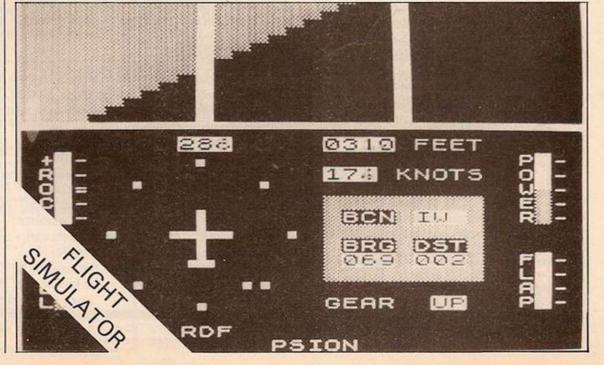
Personal Software Services — PSS — has a Krazy Kong game on the market. You have to climb a pyramid and knock out Kong's support. Obstacles to succeeding are barrels and fireballs which you must jump over on your way to Kong at the top. The graphics are good and make Kong a lively and exciting game.

What seems to be the only version of Centipede for the ZX-81 has been produced by Llamasoft. This is a very distant relation of the
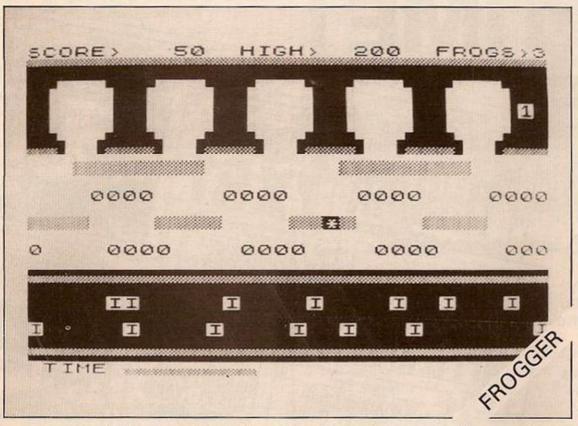
Arcade game. Rather than threading its way through a forest of toadstools the centipede is unimpeded and descends too quickly.

Two games which are deservedly best-sellers are Gulp 2 from Campbell Systems and Flight Simulation from Psion. Gulp 2 is similar in quality to the version available for the Spectrum, which means it is very good indeed. It is similar to Pac-Man in that your man has to eat the dots in the maze while being pursued by hostile monsters. What is impressive, though, is that there is a choice of fifteen different mazes; you can even have one in the shape of the letters ZX-81. The speed can be set, there is a choice of two keying systems, and the game will give you a demonstration before you begin. It is not hard

FLIGHT SIMULATOR

ZX-81 SOFTWARE

SCORE>   50   HIGH>   200   FROGS>3

FROGGER

STARQUEST

to see why this game has proved so popular.

Flight Simulation is just what the name says — a realistic and complicated simulation of flying. You have to direct your plane to the airport and then land it. There are two separate displays. The first is a cockpit with a view of the horizon and control panel. The second is a map showing your position relative to the landing strip, which enables you to navigate.

It takes quite some time to master the technique of landing since you have to co-ordinate height and speed then bank and dive to your death watching the horizon sway sickeningly through the cockpit windows. The more sophisticated will take the time to master the techniques of flying and

landing, perhaps so that they can take over when the pilots of their Boeing 707 die from food poisoning.

But software does not have to be as complex as this to be enjoyable. Psion also markets a version of Breakout called Thro' the Wall — on the same tape as their Scramble. It is just as you would expect; a bat, a ball and a wall of bricks. The simple pleasures of this game take one back to the days when a computer ping-pong machine was as awe-inspiring as the first sunrise. The uncomplicated principles of games such as Breakout might guarantee their existence in an age of ever more noisy and baroque programs.
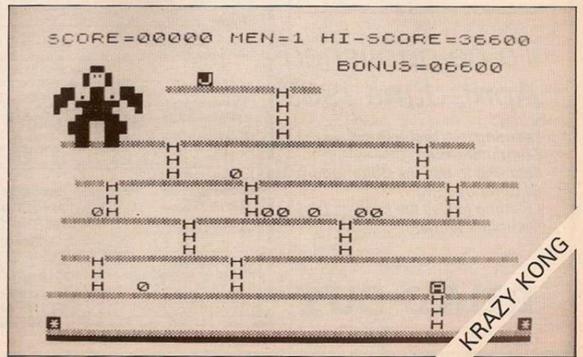
Amba Software has two similarly straightforward games; Trap and Hunt. In the first you play an opponent, or the computer, and have to block off areas of the screen by drawing lines. In Hunt, again a two-player

game, you chase numbers in a maze; as time runs out the value becomes smaller. There is something compulsive about Hunt, involving as it does, the race against time and avoiding obstacles.

Adventure games are one field where the ZX-81 does not lose much by comparison with more powerful machines. This does not mean that games for the ZX-81 are different or better; on the whole we get a tepid goulash of the same leftovers. Fantasy games from Psion offer two adventures, Perilous Swamp and Sorcerer's Island. The deep stagnant pools of the first sum up the whole world of the adventure game. The princess, the evil wizard, the slimy green bunyip, the horrible dirty troll; love them or hate them, you cannot play an adventure game without them.

At least, it seemed you could not. Dictator, from Bug-Byte, looked like a welcome exception. Here, you are President for life of the banana republic of Ritimba. In this somewhat amoral game you have to exploit the country for your own good and then make your escape before the guerillas, or other opponents, get you. However, the colourful blurb and drawings flatter only to deceive. When you settle down to be a ruthless and evil megalomaniac you will be disappointed by the dullness of the game itself. Quicksilva, though, has succeeded in finding something different. Trader — in three parts — is more of a semi-adventure game; there are the usual decisions to be made, but also tasks such as piloting your ship or finding your way through a maze.

The variety of activities and the clever graphics make this much more entertaining than the usual adventure. Your objective, of course, is to make a huge profit by trading with the inhabitants of the Meridien system, all of whom have different needs and characteristics. The Psions are immensely intelligent beings, constantly rearranging the blobs which constitute their bodies. They are con-

SCORE=00000   MEN=1   HI-SCORE=36600
BONUS=06600

KRAZY KONG

A NEW SERIES FOR FIRST TIME USERS

"**Learning to Use**" is a new series of books which introduces newcomers to the most widely used micros in the marketplace.

The books assume absolutely no knowledge about computers and the reader is shown even the most fundamental operations such as "switching on" and "loading a program". The books lead the reader through simple programming and then on to graphics, with several programs which show how to achieve pictures and even animation!

The user-friendly approach is consistent throughout the text — not only are program listings clearly shown, but in many cases, a photograph is included to show what the program looks like when actually loaded and run!

All books in the series are £5.95 (incl. postage).

Gower

To be published April–June 1983

Learning to Use the Oric 1 Computer (April)
Learning to Use the Commodore 64 Computer (May)
Learning to Use the Lynx Computer (June)

READ-OUT

temptuous of human intelligence and, before doing business with them, you will have to answer their questions correctly. With Betans you must bargain. With the robotic Alphans you can only take it or leave it. On Delta though, you can make a killing.

These junkies will pay an enormous price for your Boosterspice, a kind of inter-galactic heroin. But you have to be careful they do not mug you or your might pass out of the Meridien system, as I did, in a sordid back alley. Trader is great fun — do not be put off by the unnecessary packaging and silly booklet.

Traditional board games are also in evidence in ZX-81 software. Psion has both chess and backgammon tapes. Of the two programs backgammon is the more satisfying, since it suffers less from the graphics limitations of the ZX-81. By playing the computer you can teach yourself how to play the game. The display is clear and wrong moves are explained. Chess, on the other hand, is rather confusing; as the program notes suggest, it is a good idea to set up the game on a normal board. Another drawback is that moves are entered using algebraic notation — for example, E2-E4, but the board on the screen has no letters or numbers. If you are not familiar with the system it can take you some time to figure out what to enter. The computer's response is quick at levels 1 and 2, but on level 3 can take several minutes. At the two lower levels it plays a reasonable, though not very strong, game of computer chess.

Better value
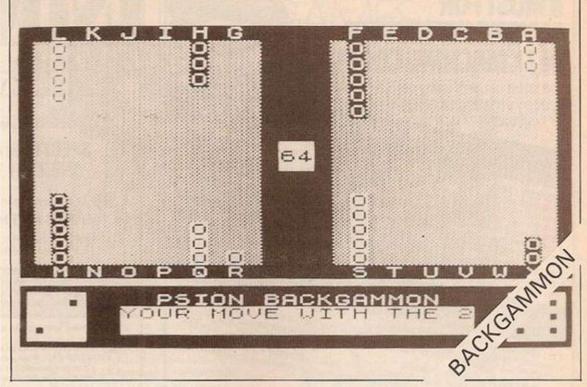
Generally, those programs that have some practical application are better value than the games software and, to my mind, more interesting. Games, after all, are not the ZX-81's forte; most people buy the machine primarily as an introduction to computing rather than as a games machine.

In particular, probably more people have gained some familiarity with machine code on the ZX-81 than on any other home micro. Few Vic owners for example know where or how to insert machine-code routines; every dedicated

ZX-81 owner, by contrast, has learned the value of a hex loader or the significance of the figure 16514.

An assembler, of course, removes the hassle of hex loading. Bug-Byte's ZXAS allows you to enter standard Z-80 mnemonics in Rem statements and also accepts labels for jump and call instructions. It takes up 5K and will assemble code at any specified location.

Assemblers are indispensable to the serious machine-code programmer and Bug-Byte's is as effective as any other available. To go with it the well-equipped programmer also needs a machine-code monitor. Two on offer are from Picturesque and Crystal Computing. These are both well-written and useful programs although the Crystal Computing monitor is perhaps the superior version. In addition to providing a range of facilities for entering, displaying, moving and testing machine code, it also supplies a disassembler and the option for saving and loading blocks of memory in any part of RAM.

Picturesque's Screen Kit 1 contains a varied

selection of machine-code routines which are embedded in Rem statements and can be used from Basic programs. These include routines for scrolling in any direction, saving and loading variables, and displaying the amount of memory left. Most of these routines have already been published in home computer magazines — or, at least, in *Your Computer* — but the collection could be recommended to newcomers.

Rather more useful is Bug-Byte's ZXTK toolkit. Again this is a suite of Rem-based routines but specifically intended as aids to program development. There are ten options which offer such features as renumbering and deleting of Basic program lines as well as facilities for moving and merging programs whenever you want.
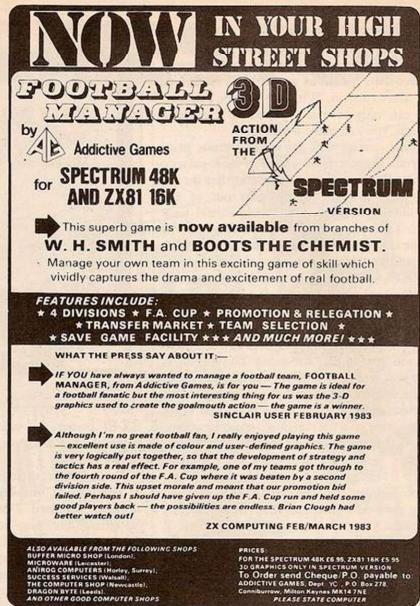
Question of ease

There is not much to choose between the two programs. The Fast One leaves around 11,700 bytes free for file space and offers a greater range of features — including an option for finding the total and average of figures within selected records. Vu-File on the other hand has slightly less file space but is easier to use.

None of these facilities would be exceptional if they were present on a larger micro; but on the ZX-81 the speed and versatility of these programs is quite an achievement — given the limitations . . .

The final word in this review, sadly, must be a critical one. Many of the programs discussed were loaded only after repeated attempts. Some programs could not be reviewed because they obstinately and persistently refused to be loaded, even when tried on different recorders. One of these brutes was, unfortunately, Gauntlett — a game I would have liked to play as it features in *Your Computer's* Top Twenty. An honourable mention has to be made here of Psion, whose programs all loaded first time. While conceding that the ZX-81 is a sensitive beast which must be approached with caution, I do not see why manufacturers cannot take a little more care with their products and help the consumer to avoid the frustration and rage which are the result of faulty loading. ∎

Company	Software	Price	Company	Software	Price
Psion,	Chess	£6.95	Crystal Computing,	Monitor	£8.95
Sinclair Research,	Flight Simulation	£7.95	2 Ashton Way,		
Stanhope Road,	Backgammon	£5.95	East Herrington,		
Camberley,	Thro' The Wall	£4.95	Sunderland SR3 3RX.		
Surrey.	Fantasy Games	£5.95			
	Vu-File	£7.95	Picturesque,	Screen Kit 1	£5.70
			6 Corkscrew Hill,	ZX-MC	£7.50
Mikro-Gen,	Tempest	£3.95	West Wickham,		
24 Agar Crescent,	Scramble	£3.95	Kent BR4 9BB.		
Bracknell,	Frogs	£3.95			
Berkshire.			DJL Software,	Frogger	£5.95
			9 Tweed Close,		
Amba Software,	Hunt	£4.95	Swindon,		
Freepost,	Trap	£4.95	Wiltshire.		
Cambridge CB3 7BR.					
			PSS,	MCoder	£7.95
Bug-Byte,	ZXTK	£6.00	452 Stoney Stanton Road,	Krazy Kong	£7.95
100 The Albany,	ZXAS	£5.00	Coventry CV6 5DG.		
Old Hall Street,	Dictator	£9.00			
Liverpool L3 3AB.	Invaders	£4.00	Silversoft,	Invaders	£3.95
			20 Orange Street,		
Quicksilva,	Galaxians	£4.95	London WC2.		
92 Northam Road,	Trader	£9.95			
Southampton SO2 0PB.			Llamasoft,	Centipede	£2.45
			49 Mount Pleasant,		
Campbell Systems,	Gulp 2	£4.75	Tadley,		
15 Rows Road,	The Fast One	£12.00	Basingstoke,		
Buckhurst Hill,			Hampshire.		
Essex IG9 6BL.					

3 GREAT EXCITING GAMES

AVENGER written by Kevin Flynn
- for the ZX Spectrum (16/48k RAM)
- for the ZX81 (16K RAM)

FEATURES
- ARCADE ACTION
- Eight different types of target.
- Rapid manouvering of ship.
- Quick firing of bombs and lasers.
- 'Intelligent' alien interceptors.
- Realistic bombing trajectory.
- Continuous display of MISSION TIME, SCORE, SHIELD & LASER TEMPERATURE.
- End of mission report.
- Five levels of play.
- Full use of sound effects and colour.

Only £5

SENTINEL written by Kevin Flynn

for the ZX Spectrum

SENTINEL is an entirely original **ARCADE STYLE** game, written in MACHINE CODE to give fast action-packed performance. The gateway to your galaxy is guarded by the Sentinel System. Only your skill and initiative stands between the advancing enemy's space fleet and the destruction of the galaxy. You have 4 space ships per station each armed with photon-cannons and guided missiles. Make use of the Force Field when ever possible.

Only £5

PROTECTOR written by Kevin Flynn

for the ZX81 (16K RAM)

Protector is written in MACHINE CODE to give an **arcade style** all action game. Using the Sentinel's laser you must defend the planet Zargon from the aliens which are bombing the planets vital installations. On screen Alien score, your score, number of shields left and number of Laser shots. Watch out for the Damacles Missile which may be detonated at the alien commander's discretion.

Only £5

Other Abacus Games

BODYBUILDER: VIC-20 (6½K RAM)
I don't know whether I'm coming or going. So many doors - so many monsters chasing me. Somebody must have my head....my arm....my leg....

£4.50

FOR THE DRAGON 32 and VIC-20 (16K RAM)
INTO THE LABYRINTE: Can you escape the curse of the sorcerer SKAROTH?
A gripping ADVENTURE game with 38 locations to explore, 40 objects to find and manipulate. There is a limit to the number of moves you can make before the curse kills you. At least 13 ways to lose, but only one way to win......frustration guaranteed.
- also 13 magical events

Only £5.99

Now available For the ORIC 1

● For the younger spectrum user

SPECTRUM GAMES PACK 2

NINE LIVES:
Catch the mice before the dog catches you. Six levels of play.

BULL TABLE:
Do your sums quickly before the Bull upsets your table. Seven levels of play.

CANDYMAN:
Race your men through the moving maze. A great game for two players.

TARGET:
All the fun of the fair.

All four games on one cassette for £4.50

DOMAIN: (Spectrum)
Probably the best version of **Kingdoms** you are ever likely to see.

£4.95

ESP ZX81 (16K RAM)
An exploration of INNER SPACE. Undertake a serious investigation of your own Extra Sensory Perception. Test your family and friends - an educational and scientific experience. **ESP Package** includes **two** 16K programs with commentary.

£4.95

SPECTRUM GAMES PACK 1
DESTROYER: Listen to the beeps of the sonar to locate and destroy the submarines before they sink the merchant ships.
ICEBERG: Steer your icebreaker through thickening pack ice to pick up survivors.

BATTLE: Destroy missile sites while avoiding mines and the computerised enemy tanks that are out to get you.
ANDROID PIT RESCUE: Rescue the miners before they are trapped in the flooding mines.
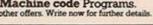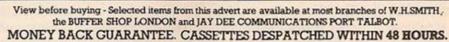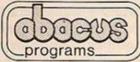
All four on one cassette at the incredible price of **£4.95**

■ **WANTED** original **Machine code** Programs. ■
We guarantee to better all other offers. Write now for further details.

MICROMUSIC: VIC-20 (6½K RAM)
No previous knowledge of music is necessary to use this entertaining and educational program. Create music speedily and easily using standard notation. A must for the budding maestro.

£4.50

ABACUS CONTROLLER

Developed to eliminate tedious swapping of plugs when LOADING or SAVING programs on cassette.
ZX SPECTRUM CONTROLLER: Single switch selection of SAVE, LOAD & AMP modes. Built in amplifier and loudspeaker boosts Spectrum sound output. Price £14.98
ZX81 CONTROLLER: Single switch selection of TALK, SAVE, CUE & LOAD modes. Built in microphone/speaker for fast and reliable program naming and cueing. Price £9.98
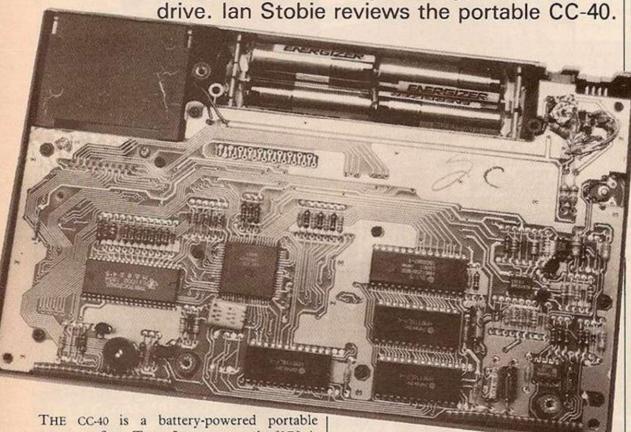
"Of all the Spectrum hardware items I have reviewed, this one most impressed me and is the only one I shall always use."
Quote from PCW.

View before buying - Selected items from this advert are available at most branches of W.H.SMITH, the BUFFER SHOP LONDON and JAY DEE COMMUNICATIONS PORT TALBOT.
MONEY BACK GUARANTEE. CASSETTES DESPATCHED WITHIN 48 HOURS.
All prices inclusive of p&p. Send cheque or PO to:

abacus programs

186 St. Helens Avenue, Swansea, W.Glam. Tel:(0792) 50282

Abacus Programs

TEXAS

Compact Computer is the first of a new wave of micros from Texas which have been designed to use a range of miniaturised peripherals including a stringy-floppy tape drive. Ian Stobie reviews the portable CC-40.

THE CC-40 is a battery-powered portable computer from Texas Instruments. At £170, it is in the same price range as the Dragon and the 48K Spectrums and Orics — but that is its only similarity. The Compact Computer will never know the satisfaction of crushing a human at space invaders; it is a machine with a serious purpose.

Up until now Texas has taken care of the home market to its own satisfaction with the TI-99/4A. The Compact Computer 40 is an attempt to get into the booming portable market, currently dominated by names like Epson, NewBrain, Casio and Sharp.

The CC-40 weighs a little more than one pound and measures 9.25in. wide by 5.75in. high by 1in. thick — about the size of a big paperback. It can run for 200 hours on four AA size alkaline batteries and it has a built-in 31-character single-line liquid crystal display and good-quality calculator-style keys in a QWERTY layout.

Cartridge slot

Inside, the CC-40 is built around a Texas own-brand eight-bit CMOS chip, the 70C20. It comes with 34K of ROM containing a comprehensive Basic, and 6K of RAM extendable to 18K internally. A cover at the top edge of the machine to the right of the display pulls off to reveal a memory cartridge slot, and a further 16K of RAM can be fitted here, giving an absolute maximum of 34K RAM. Program cartridges also fit into this slot.

Texas expects great things for the new

peripherals designed for the CC-40. These include a battery-powered four-colour printer/plotter of the Sharp type, a fast "Wafertape" continuous loop cassette drive like Sinclair's Microdrive, and an RS-232 interface unit for connecting up to big printers and, potentially, other computers.

A Modem and a black and white TV interface are also scheduled for release later in the year. The CC-40 system is clearly aimed at people using computers for their work; but this could include students of scientific or engineering subjects as well as professional users.

The whole machine is well made. For instance, along the bottom of the machine runs a recessed stand, which can be pulled down to tilt the keyboard to a nice angle for typing. There will be no need for entrepreneurs to offer adaptor feet at £4 a pair.

The casing itself is made from a silvery-grey plastic, except for the numeric keypad, a few of the other keys and the area around the screen, which are highlighted in a darker grey.

At the back of the machine is a slot for the optional AC adaptor which costs £14.95, and the Hex-bus connector, into which all the peripherals will fit.

There were no problems with the keyboard. The calculator-like keys feel comfortable and give tactile feedback. The layout is fairly close to the standard QWERTY pattern, but the Enter key, the Texas equivalent of Return, is

rather annoyingly positioned where you might expect the right-hand shift key to be.

Given the size of the CC-40 the keys are obviously smaller than on a standard typewriter, but they are easy to use and larger than on the Sharp PC-1500 for instance, a machine with virtually the same price and an obvious competitor.

The CC-40 comes with a plastic keyboard overlay, which among other things has the principal Basic keywords on it. Basic keywords can either be typed in full in the normal way, or by using the function key then hitting the appropriate single key given on the overlay, the machine can be made to produce them Sinclair-style on the display. This is a useful time-saving feature, although the overlay itself might soon be lost.

Program scrolling

The display is a single line of 31 characters, but this can be used as a window on to an 80-character line. The arrow keys above the keypad can be used to scroll around horizontally within the line, or to step vertically within through a program listing. With a little practice the small display does not seem such a limitation, although for developing long programs the printer at £149.95, would be essential.

On the side of the machine to the left of the display is a knurled wheel. Turning this adjusts the display angle electronically, so you can aim it at yourself to get the best contrast.

The characters are formed on a five-by-eight matrix, and like all LCD displays are much less tiring to watch than a TV screen. The lack of colour confirms that this is a working machine, rather than an entertainment device.

On both sides of the actual text line in the display one finds indicators showing battery-low warnings, upper-case lock on, Function, and which units the trigonometric functions are currently returning value in.

The machine can be used as a calculator. It works exactly the same as Basic in immediate mode, but on the CC-40 it is not necessary to type Print before a calculation. You just key in 8*63 and then hit Enter, and it gives you the answer.

The up-arrow key, used with Control held down, functions as a playback key. Hitting it brings back the last line entered on to the display, where it can be edited, and the calculations redone. This saves time if you have entered a long complex calculation and want to try it out with a few different values.

For anything more complex there is Basic, and the CC-40 has a very full Basic. Again, it is practically oriented. There are no sound commands to support the small internal beeper, only the feeble Beep, but this is sufficient for attracting attention.

The numeric precision is good — 10 displayed significant digits — but the machine is not very fast. Battery-powered CMOS machines are inherently slow.

If your foreign languages need brushing up try,   Call Setlang 1 This puts out all the error messages in German — *Variable nicht definiert* — and that kind of thing. Other languages are available on ROM cartridges.

The Basic has all the usual commands of

CC-40 REVIEW

CONCLUSIONS

- The CC-40 is a well-made, battery-powered portable machine intended mainly for serious use. It will be good for field-work, travel, and as a sophisticated calculator replacement.
- The machine has no colour capability, and compared to a home-based micro limited maximum memory and a slow processor speed. It is not a games machine.
- The CC-40 is aimed at professional and technical users. It has a full and powerful Basic.
- Its main rivals will be the Sharp PC-1500, and perhaps even the luxury £700 Hewlett-Packard HP-75C.
- For serious users the range of mostly battery-powered peripherals will make the machine very attractive. The CC-40 really comes into its own as the central unit in a system.

Microsoft-style Basic. It includes If . . . Then . . . Else, On . . . Goto/Gosub/Error, Print Using to format output conveniently, and various file control commands for use with the Wafertape, like Open, Print#, Input#, and Verify. Three-dimensional arrays are allowed, and long variable names.

Strings can be up to 255 characters long. String handling works differently to most home micros. Instead of Left$, Mid$, Right$ the CC-40 has a Seg$ command for extracting substrings, and a Pos command to find them.

Another addition is the way subroutines are handled. Gosub is there and can be used in the normal way, but there are several more powerful commands allowing you to set up sub-programs. These are called with a Call statement and an optional set of parameters.

The 34K ROM contains a number of useful sub-programs. Call Debug gets you into a machine-code monitor, with all the usual Examine, Move, Copy and start Executing commands, plus the ability to set Breakpoints.

The CC-40 manual which describes all this is extremely good. It is a 300-page professional manual, not just a home micro manual.

Ready-written software will be available on cartridges and Wafertapes. Statistics, Finance, Maths and Electronics cartridges should be available immediately, with Perspective drawing, business graphics and a few others hot on their heels according to Texas. These all cost £59.95. Two games cartridges cost nearly £40, and the Assembler nearly £125, so software is not cheap.

The Wafertapes will be cheaper, just under £20. Again the announced titles look fairly professional — things like Elementary Dynamics, Pipe Design, Profitability Analysis. Texas hope to find third-party software suppliers to contribute additional titles. ■

BENEATH THE STREETS of Cambridge he waits for the call. A spear hangs from the ceiling, a sword from the wall. Any moment the red phone could ring and send Captain Spectre's Flying Squad and their electronic ghost trap out on the trail of another ghoul.

The scene sounds straight out of *The Avengers* but Captain Spectre's quarry is more elusive than any that Emma Peel or John Steed tracked down. His mission is to capture for the first time on film denizens of another world — spirits, poltergeists, ghosts. His only weapon is a 16K Spectrum linked to an array of sensors, lights, recorders and cameras.
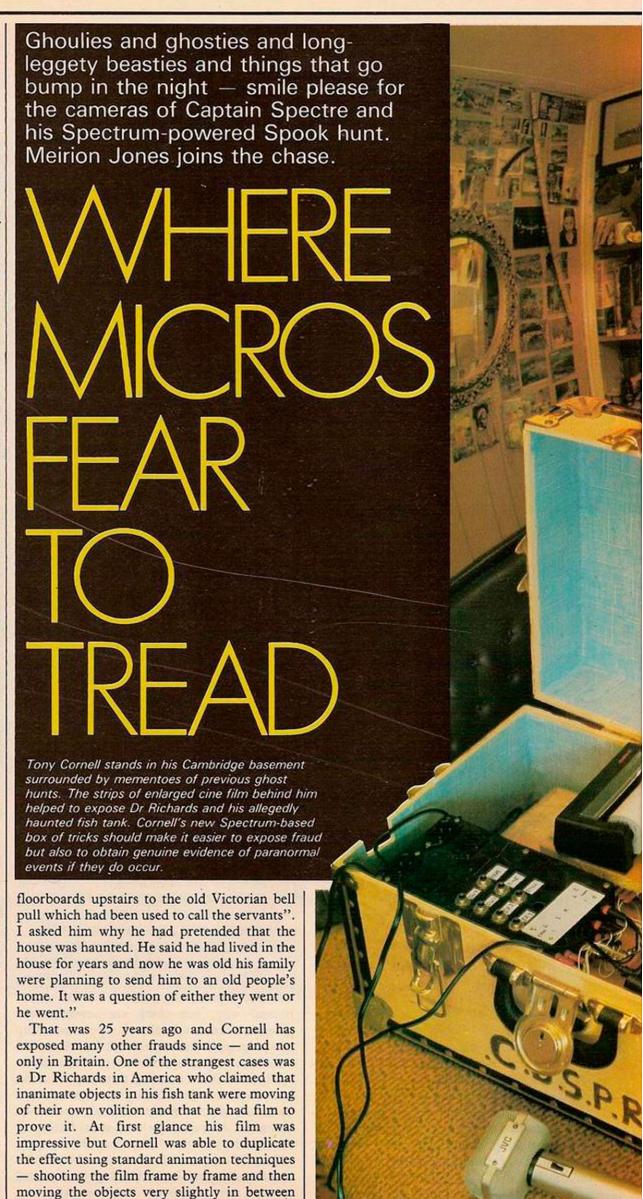
Captain Spectre — or Tony Cornell as he would rather be known — is fully aware of the problems of capturing spirits on celluloid. As Treasurer of the Society for Psychical Research he has been able to leaf through the thousands of supposed ghost photographs sent to the SPR since it was set up a century ago to investigate paranormal phenomena. He believes that there is "not one photograph in the archives that we would today accept as genuine."

Traditional methods of investigating haunted houses or poltergeists, are unlikely to produce scientifically acceptable results. An observer who has been left for the night in some gothic manor is quite likely to be seeing things by the early hours of the morning whether they are there or not and if anything should happen the observers may not be able to turn on recorders or take a photograph in time.
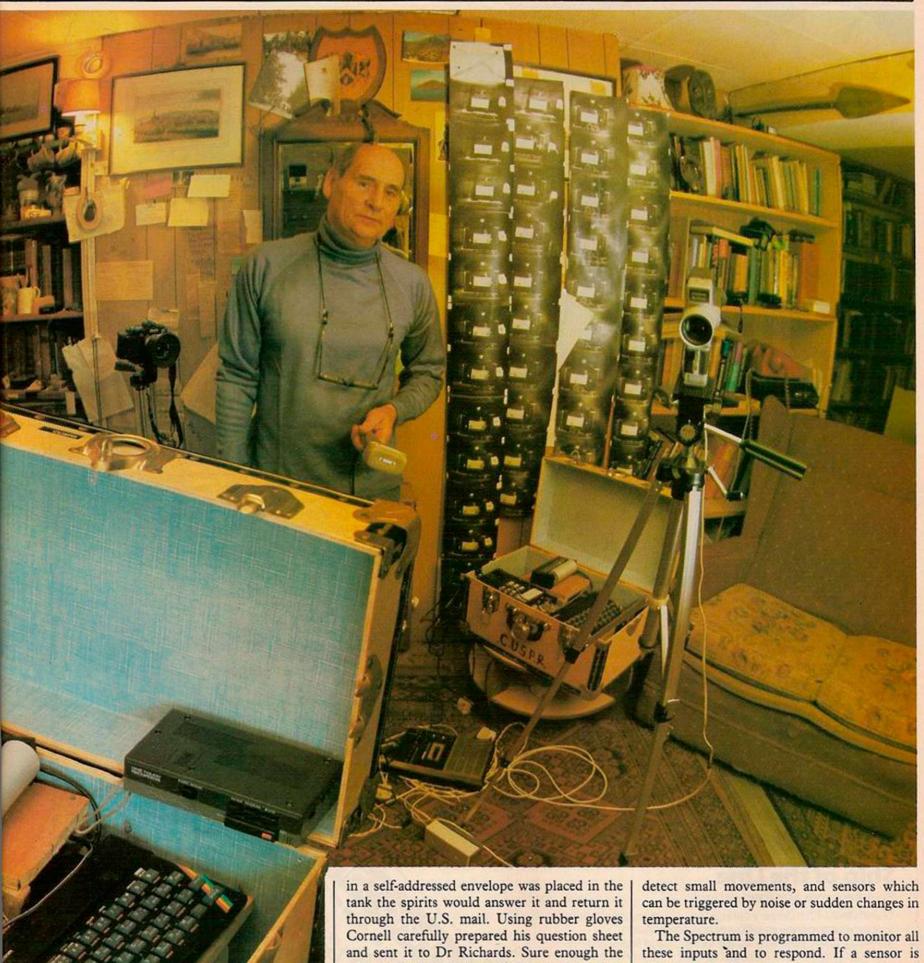
An automatic observation station with cameras and recorders that can be instantly triggered by movement, temperature changes or sound and can be left to monitor a site for days at a time is far more likely to produce useful results. It is also less likely to be taken in by the many deliberate frauds and practical jokes which have often taken in investigators of the paranormal. Sherlock Holmes's creator Sir Arthur Conan Doyle who joined the SPR after pushing his all-seeing detective off the Reichenbach Falls remained convinced to the end of his life by a film purporting to show fairies dancing at the bottom of his garden. Last month — 60 years later — one of the fairies, Mrs Elsie Hill, finally admitted that it had all been a hoax.

Alan Cornell's first investigation of a poltergeist convinced him of the dangers of taking things at their face values. He had gone to an erie-looking Victorian house in North London where a mysterious crashing and banging noise in empty rooms had convinced a family that the house was haunted. The father, mother and three children had all moved out leaving only Grandad who refused to be scared off. On the evening that Cornell called in the old man was sitting in a corner by the fireplace. Cornell sat down and shortly afterwards he heard "thumps and bumps and crashes and bangs upstairs". A check of the upstairs room yielded no clues — there was nobody else in the house apart from the old man downstairs in his armchair. When the noise started again Cornell noticed out of the corner of his eye that Grandad was moving his shoulder.

"He had connected up two wooden balls and a heavy piece of lead under the

Ghoulies and ghosties and long-leggety beasties and things that go bump in the night — smile please for the cameras of Captain Spectre and his Spectrum-powered Spook hunt. Meirion Jones joins the chase.

WHERE MICROS FEAR TO TREAD

Tony Cornell stands in his Cambridge basement surrounded by mementoes of previous ghost hunts. The strips of enlarged cine film behind him helped to expose Dr Richards and his allegedly haunted fish tank. Cornell's new Spectrum-based box of tricks should make it easier to expose fraud but also to obtain genuine evidence of paranormal events if they do occur.

floorboards upstairs to the old Victorian bell pull which had been used to call the servants". I asked him why he had pretended that the house was haunted. He said he had lived in the house for years and now he was old his family were planning to send him to an old people's home. It was a question of either they went or he went."

That was 25 years ago and Cornell has exposed many other frauds since — and not only in Britain. One of the strangest cases was a Dr Richards in America who claimed that inanimate objects in his fish tank were moving of their own volition and that he had film to prove it. At first glance his film was impressive but Cornell was able to duplicate the effect using standard animation techniques — shooting the film frame by frame and then moving the objects very slightly in between shots.

Dr Richards then claimed that if a question

in a self-addressed envelope was placed in the tank the spirits would answer it and return it through the U.S. mail. Using rubber gloves Cornell carefully prepared his question sheet and sent it to Dr Richards. Sure enough the answer came back through the post but when fingerprint powder was dusted over the sheet it revealed that Dr Richards' spirits had human finger and palm prints.

Although Cornell soon realised that automatic continuous surveillance equipment would make psychic research much easier, until recently the cost of such a system would have been prohibitive. Cheap home computers changed all that. The box of tricks at the heart of Cornell's monitoring system contains a 16K Spectrum, a Sinclair printer, a DCP Interspec, and a series of relays. An array of sensors can be connected up to the box. These include two infra-red beams and a doppler-shift ultrasonic alarm to detect movement, circuit breakers to detect small movements, and sensors which can be triggered by noise or sudden changes in temperature.

The Spectrum is programmed to monitor all these inputs and to respond. If a sensor is triggered the Spectrum turns on lights and runs a cine camera for 20 seconds. It also turns on a tape recorder for one minute and takes six still photographs with a motor-driven camera. The program also helps set up the equipment by checking through stage by stage with the operator that everything has been correctly connected. The Spectrum prints out a record of what responses were triggered and when. Once it has been set up the equipment can function for several days without attention.

The first test for the new box of tricks came this Christmas. A 200 year-old chest of drawers in an antiques warehouse had been at the centre of some strange happenings. For

Psychic researcher Tony Cornell and his box of tricks together with the still camera, cine camera, and infra-red and ultrasonic detectors which plug in.

weeks every morning when the staff came in the drawers would be hanging out at strange angles even though they had been pushed in when the staff had locked up the night before. Eventually Tony Cornell was called in. He and the Cambridge University Society for Psychical Research, CUSPR, staked out the storeroom the chest was in with their box of tricks. The response was triggered eight times but the drawers had not moved and there was nothing unusual on the film. The equipment may have been triggered by a spike in the mains current so a mains smoother has subsequently been fitted. Since then there have been no further strange happenings with the chest which now sits uncomfortably in Tony Cornell's basement with two tribal sculptures eyeing it suspiciously from across the room.

Cornell is cynical about haunted houses — he believes that ghosts may be in the eyes of the beholders once they have been told that a particular place has a spooky reputation. Once he and several members of the SPR were invited to the Ferryboat Inn in Huntingdon to witness an apparition by the White Lady who was supposed to appear every March 17. "The landlord gave us dinner, there were hundreds of sightseers, Americans everything. No-one had ever seen the White Lady — it was terrible. Some months later I saw it in a Sunday magazine described as 'The fifth most haunted house in Britain'."

Because of his own experience he is less cynical about poltergeists. At Hanneth Hall in Wisbech, Cambridgeshire, he was among a team of investigators who tore the place apart looking for any normal explanation of the bumps and thumps and objects flying around which they experienced there.

At one time it was thought that such effects might be caused by vibration perhaps from underground streams or seismic activity. Cornell's own experiments convinced him that was not the case. He obtained permission to subject four cottages which were scheduled for demolition to intense vibration. Although the tests were continued to the point when the cottages were about to collapse they were unable to duplicate the effects attributed to poltergeists.

Spectrum telepathy

Richard Loosemoore, who built Cornell's box of tricks for just over £200 — the cameras and sensors are borrowed as and when necessary — is hoping to use a 48K Spectrum for more psychic research. He is conducting research into Artificial Intelligence — in particular cognitive psychology. But his own interest is telepathy. He believes that "there is no question about the proof of telepathy — it is just a question of what the mechanism is."

In Loosemoore's experiments two volunteers sit in separate rooms and one attempts to transmit his or her emotional state — agitation, relaxation or anxiety — to the other by telepathy. Using the same principle as a lie detector the observers measure the Galvanic Skin Response — changes in skin conductivity — of both volunteers. If they really are able to transmit emotions by telepathy then the observers should be able to detect simultaneous changes in the GSRs of both volunteers.

In the past Loosemoore has had to borrow expensive purpose-built machines when he can to process and evaluate the data. The CUSPR has now given him £300 which he hopes will allow him to build a new system around a 48K Spectrum which will make it possible to carry out far more tests.

Tony Cornell has found the recent publicity for his investigations of the paranormal to be a mixed blessing. He never knows who is going to be on the other end of the line when the red phone goes. "I could be called out by two raving nutters — there is a lot of cranky stuff in this field". The shelves of Cornell's basement contain everything from Nostradamus's prophesies of doom and destruction to Eric von Daniken's tales of prehistoric spacemen.

Cornell suspects that "some as yet little known human faculty" may be responsible for psychic phenomena. His failure to catch a spirit red-handed opening the 200-year-old chest of drawers has not led him to believe that the spirits have something against being photographed. "After all" he says sounding more like a social worker than a psychic researcher "they are only doing it to draw attention to themselves."

To turn those dreams of machine-code programs into living facts, take Phil Holliday's advice on assemblers and monit

ASSEMBLERS AND monitors are designed to make the process of machine-code programming as simple as possible. The input to an assembler — the source code or program — consists of lines of assembly-language instructions, with optional labels, comments, assembler directives and assembler commands. Labels identify a particular point in the program. Comments enable you to document your program.

Assembler directives are pseudo-instructions — not Z-80 mnemonics — which either provide parameters for operation of the assembler, or instruct the assembler to store values directly in memory. Commonly provided directives include Org — short for origin — which supplies the starting address of the machine code to the assembler and Equ which assigns a value to a label. DefB, DefW and DefM insert a byte, word or string respectively at the current address.

Unlike assembler directives, commands have no effect on the object code produced by the assembler but instruct the assembler to perform certain operations, such as to produce a heading in the listing, or change the address in the listing from hexadecimal to decimal.

The output from the assembler is the hexadecimal machine code — object code — which is loaded directly into memory, and optionally a listing which has the addresses and op-codes added to the source file.

The assemblers reviewed used one of two methods to enter the source code. Either the source code is entered in Rem statements in Basic, or an integral editor is supplied as part of the assembler package. Normally these editors use line numbers, just as in Basic, but with an auto-line number, renumber and block-delete facility. Since most of the time spent sitting at the keyboard is actually spent using the editor, either just entering source code, or else modifying it in the debugging phase, it is most important to have an editor which is quick and easy to work.

Finally having entered and assembled a source file, it must be run and debugged. A monitor is required for this. Some assemblers

do contain a simple set of monitor commands but, in general, a more comprehensive monitor is required. In particular, commands are required which allow registers and memory to be inspected and modified, and to run the program from a given address with break-points set to terminate the run at a given address or addresses. Other necessary facilities are the ability to relocate the assembled object code — it may actually need to be run in the area occupied by the assembler for instance — and a disassembler would be required if the assembler does not produce a listing of the assembled program.

In ACS Software's Ultraviolet Assembler, the Basic editor is used to enter source code into Rem statements. A dummy Rem statement, long enough to hold the assembled code, must be created as the first line of the program. The source code is then assembled by the Ultraviolet assembler and a listing is sent to the screen, with different classes of op-code being printed using different paper colours. The listing can optionally be sent to the ZX Printer. All numbers must be entered in decimal, and multiple statements can be entered in one Rem line. The Infrared disassembler is also available from ACS and produces a similar colourful output to the Ultraviolet assembler.

The Ultraviolet Assembler — from Artic Computing contains a built-in monitor and only runs on the 48K Spectrum. The source code is automatically stored in a Rem statement at line 2, and the object code is put into a Rem statement in line 1. The editor is fairly comprehensive. It does not use line numbers, like all the other products reviewed here, but has a screen editor mode in which the cursor can be moved up and down to point at lines. Lines can be inserted or deleted at the cursor position. A line-editor mode is also available allowing text to be modified.

The assembly is initiated from the built-in monitor, which also allows the assembled program to be run. Most of the normal monitor functions are included, allowing memory and registers to be inspected and modified and blocks of memory to be moved, but it does not have a breakpoint facility which is essential for debugging. Also it has no disassembler, which is vital since the assembler does not produce a listing of the assembled code.

One very useful feature of this product is a memory status report at the bottom of the screen showing how much memory is occupied by the source, object, labels and Basic program and how much is still free. The documentation is good, providing clear instructions and hints and examples to the user.

The Aspect 4.2 Editor-Assembler by Bug-Byte proved to be a very disappointing program. The editor is most unfriendly. It has an invisible cursor, which makes entering and modifying the source code very difficult indeed. Text cannot be inserted or deleted without first deleting all characters from the end of the line back to the required point. The labels can be of any length but they are cut to five characters length in the listing. The symbol table space is fixed at approximately 450 bytes, which may be very limiting in some
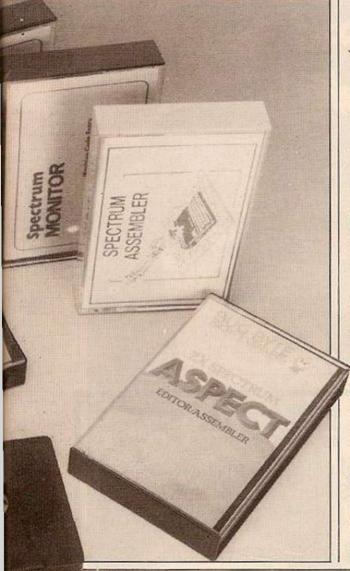
applications. There is no way of outputting the listing or symbol table to the printer, which makes debugging very difficult. The documentation is barely adequate.

The assembler from CP Software is very limited. It is written in Basic and assembles the code at a painfully slow rate. The source code is entered into Rem statements. Labels are limited to three characters in length, and the only assembler directive provided is DefB.

The Zeus assembler from Crystal Computing is a well thought-out program. It has a true screen editor. Text may be displayed and modified by moving the cursor to the appropriate place in a line on the screen. It also has a good auto-line number, renumber and block-delete facility. This editor was easy and pleasant to use. The assembler does not produce a listing, but the source code and symbol table can be sent to the printer or screen. A built-in monitor allows the assembled object code to be run, and has other useful features.

However, a separate and more comprehen-

List of suppliers	Price
ACS Software, 7, Lidgett Crescent, Leeds LS8 1HN.	
Ultraviolet assembler	£7.50
Infrared disassembler	£6.75
Artic Computing, 396, James Reckitt Avenue, Hull, North Humberside.	
48K Spectrum Assembler	£9.95
Spec Bug monitor	£6.95
Bug-Byte, 100 The Albany, Old Hall Street, Liverpool L3 3AB.	
Aspect assembler	£9
CP Software, 17 Orchard Lane, Prestwood, Buckinghamshire HP16 0NN	
Spectrum assembler	£4.95
Crystal Computing, 2 Ashton Way, East Herrington, Sunderland SR3 3RX.	
Zeus assembler	£8.95
Monitor and disassembler	£8.95
Hisoft, 60 Hallam Moor, Liden, Swindon SN3 6LS.	
DevPac	£12
Picturesque, 6 Corkscrew Hill, West Wickham, Kent BR4 9BB.	
Spectrum editor-assembler	£8.50
Spectrum monitor	£7.50

RUM ASSEMBLERS AND MONITORS

sive monitor is available from Crystal Computing which allows breakpoints to be set and the program to be run and disassembled. The documentation supplied is very good. The manual starts with an example of a machine-code program and the user is taken step by step through the process of editing and assembling it.

The most comprehensive product reviewed was DevPac by Hisoft. The assembler-editor — Gens — and monitor — Mons — are two separate programs but are sold as one product — DevPac. The editor is an extremely comprehensive line editor. It takes a little time to learn to drive it but it is well worth the effort. It includes auto-line number, renumber and block delete.

The line-editing commands available are too numerous to list here. Some examples are commands to delete or insert characters, delete all characters from the cursor to the end of the line, reload the edit buffer from the text and many others.

The assembler is very fast and the only one reviewed with the powerful conditional assembly feature If-Then-Else. It also allows expressions of decimal, hex, binary, character constants and labels with operators of addition, subtraction, integer multiplication and division, logical And, Or and Xor, and mod operators. There are many assembler commands available which can be inserted into the source code. It is possible to enable and disable the listing and printer, and to list the location counter in hexadecimal or decimal

with the appropriate commands. Options can also be selected at assembly time, enabling the object code and listing to be turned off, and to direct the listing to the printer.

The Mons monitor provides all the usual commands one would expect. Registers and memory can be inspected and modified, and memory can be moved. It also contains a disassembler which inserts labels into the disassembled listing. The object code can be executed, and break-points set, but what sets this monitor apart from all the rest is that it can be single-stepped through the program, with a disassembly of the instruction executed displayed. Commands are available which allow single-stepping to continue after a jump or call. This, like the assembler, is a most comprehensive tool. The DevPac will appeal to the true machine-code freak who wants to write a lot of software. The documentation is first class.

The first thing that you notice about this neat product, the Editor Assembler by Picturesque, is that instead of the normal 32 columns on the screen it gives you 40. This is a very real advantage for listing the assembled source program. The whole package is very friendly and easy to use. The line editor is quick and simple to use.

The line is automatically set up with tabs at the required places. This is the fastest editor to use of all those reviewed. The only criticism is that characters cannot be inserted within a line without first deleting characters from the end of the line back to the required point.

The assembler is fast and the listing produced is very easy to read because of the 40-column line. The listing can be turned off or sent to the printer at assembly time. An extremely useful function, especially for those with 16K machines, is that once a module of source code has been assembled it can be cleared from the text buffer whilst retaining the symbol table. Thus a large program can be split into smaller modules and still be assembled, even with label references which cross module boundaries. To help this there is an assembler directive, DefL, available which allows a label value to be redefined within a program. Source and object code can be saved to tape from within the assembler.

A monitor program, which has all the normal monitor commands and includes a disassembler, is also available and documentation for both products is very good.
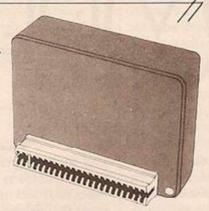
The Picturesque and Crystal Computing products can be highly recommended for those who are just about to start programming in machine code, whilst for those who are already familiar with the techniques then the DevPac is most highly recommended. ∎

Table 1.							
Company	ACS	Artic	Bug-Byte	CP	Crystal	Hisoft	Pictures-que
Product name	Ultra-violet	Assem-bler	Aspect	Assem-bler	Zeus	Gens	Assem-bler
16K	Yes	No	Yes	Yes	No	Yes	Yes
48K	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Label size	Any	Any	Any	3	14	6	5
Hex	No	Yes	Yes	Yes	Yes	Yes	Yes
Decimal	Yes	Yes	Yes	No	Yes	Yes	Yes
ORG	Yes	Yes	Yes	No	Yes	Yes	Yes
EQU	Yes	Yes	Yes	No	Yes	Yes	Yes
DEFB	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DEFW	Yes	Yes	Yes	No	Yes	Yes	Yes
DEFM	Yes	Yes	Use DB	No	Yes	Yes	Yes
DEFS	No	No	Yes	No	No	Yes	Yes
ENT	No	No	No	No	Yes	Yes	No
Cond assembly	No	No	No	No	No	Yes	No
Opt assembly	No	No	No	No	No	Yes	No
Save to tape	Basic	Basic	Basic	Basic	Basic	Yes	Yes
Error reports	5	6	4	5	10	17	8
Output to printer	Yes	Yes	No	Yes	Yes	Yes	Yes
Listing	Yes	No	Yes		No	Yes	Yes
Editor	Basic	Screen	Line	Basic	Screen	Line	Line
Documentation	Sparse	Good	Poor	Poor	Good	Excellent	Excellent
Monitor included	No	Yes	No	No	Yes	No	No
Monitor separate	No	No	No	No	Yes	Yes	Yes

THE Midland Computer Fair

Personal computers
Home computing
Small business systems

BINGLEY HALL, BIRMINGHAM 28-30 APRIL 1983

Thurs & Fri 10am – 6pm
Saturday 10am – 5pm

Presented by **YOUR COMPUTER** **Practical Computing**

This event is the first of its kind in the Midlands, and gives you the opportunity to see and compare the enormous range of personal and home computers, small business systems, microcomputers, software packages, cassettes and scores of the very latest computer games – try them for yourself – decide how much, or how little it takes to build up your own personal computer system.

HOW TO GET THERE

BY RAIL Concessionary rail fares are available direct to New Street Station. Further details are available from, British Rail Travel Centre, New Street Station, Birmingham B2 2QA Tel: 021 643 2711.

BY BUS every few minutes from New Street, Corporation Street, Colmore Row and Bull Ring.

BY CAR Bingley Hall is situated close to the city centre and is within easy access to the M1, M5 and M6. Bingley Hall will be road signed by the AA.

Admission prices – Adults £2.00 Children under 16 and O.A.P's £1.00. Party Booking: For groups of over 20 people – adults £1.50 children 75p, (plus a free ticket per 20 sold for the organiser or teacher).

- -

Please send . . . . . adult . . . . . child tickets Enclosed remittance £ . . . .

NAME . . . .

ADDRESS . . . .

. . . . TELEPHONE NO . . . .

- -

Send to Midland Computer Fair Ticket Office, I.P.C. Exhibitions, Surrey House, Throwley Way, Sutton, Surrey Tel: 01-643 8040

THE Midland Computer Fair This coupon is worth 50p off any full price ticket at the door **50p OFF**

```
10 REM**********************
20 REM*****BATTLE TANKS*******
30 REM*********BY***********
40 REM*****V.COCKETT ESQ. *****
50 REM***********************
60 PCLS:CLS0:DIMA(7,4)
70 DATA32,165,173,154,160,33,173,168,171,153,1,160,172,174,147,-31,150,174,162,1
67,-32,149,160,170,171,-33,173,153,171,161,-1,167,156,160,163,31,164,174,150,167
80 FORT=0TO7:FORT1=0TO4:READA(T,T1):NEXT:NEXT
90 GOSUB610
100 GOSUB370
110 L=&H4E1:R=&H4F7:M=32:M2=32:Y=1
120 R1=165:R2=173:R3=154:R4=160:L1=R1:L2=R2:L3=R3:L4=R4
130 PRINT@33,HL;:PRINT@60,HR;
140 IFHL=10 ORHR=10 THEN1080
150 LR=JOYSTK(0):LM=JOYSTK(1):RR=JOYSTK(2):RM=JOYSTK(3)
160 PLAY"V3T25503CFC
170 IFLM<5 THENX1=L:C=128:XL=1:GOSUB600
180 IFRM<5 THEN X1=R:C=128:XR=1:GOSUB600
190 IFPEEK(L-M)>239 ORPEEK(L-1-M)>239 ORPEEK(L-31-(2*M))>239 ORPEEK(L-32-(2*M))>
239 THEN210
200 IFXL=1 THENL=(L-M)-((L<&H400)*512)+((L)&H615)*512):GOTO220
210 IFG=3 THENX=L:HL=HL-1:GOSUB1030:C=128:X1=R:GOSUB600:GOSUB1140:GOTO950
220 GOSUB350
230 IFPEEK(R-M2)>239 ORPEEK(R-1-M2)>239 ORPEEK(R-31-(2*M2))>239 ORPEEK(R-32-(2*M
2)Y>239 THEN250
240 IFXR=1 THENR=(R-M2)-((R<&H400)*512)+((R)&H615)*512):GOTO260
250 IFG=3 THENX=R:HR=HR-1:GOSUB1030:C=128:X1=L:GOSUB600:GOSUB1140:GOTO950
260 GOSUB360
270 P=PEEK(&HFF00):IFP=125 ORP=253 THENX=(R-(2*M2)):R5=X:F=M2:GOSUB420
280 IFP=126 ORP=254 THENX=(L-(2*M)):L5=X:F=M:GOSUB420
290 IFY=0 THEN110
300 M1=M:IFM=1 ORM=-32 THENM1=-M1
310 M3=M2:IFM2=1 ORM2=-32 THENM3=-M3
320 IFLR<5 ORLR>59 THEN AR=(AR+(1*SGN(32-LR)))+((AR=0 ANDLR>59)*-8)+((AR=7 ANDLR
<5)*8):M=A(AR,0):L1=A(AR,1):L2=A(AR,2):L3=A(AR,3):L4=A(AR,4):GOSUB350
330 IFRR<5 OR RR>59 THEN BR=(BR+(1*SGN(32-RR)))+((BR=0 ANDRR>59)*-8)+((BR=7 ANDR
R<5)*8):M2=A(BR,0):R1=A(BR,1):R2=A(BR,2):R3=A(BR,3):R4=A(BR,4):GOSUB360
335 XL=0:XR=0
340 GOTO150
350 POKEL,L1:POKEL+1,L2:POKEL-31,L3:POKEL-32,L4:RETURN
360 POKER,R1:POKER+1,R2:POKER-31,R3:POKER-32,R4:RETURN
370 FORT=0TO4:POKE&H44D+T,255:POKE&H5AD+T,255:NEXT
380 FORT=0TO192 STEP32:POKE&H484+T,255:POKE&H49B+T,255:NEXT
390 FORT=0TO64 STEP32:POKE&H4CA+T,255:POKE&H4D5+T,255:NEXT
400 POKE&H46D,255:POKE&H471,255:POKE&H483,255:POKE&H49C,255:POKE&H4EF,255:POKE&H
543,255:POKE&H55C,255:POKE&H58D,255:POKE&H591,255
410 RETURN
420 FORT2=1TO12:Y$=STR$(13-T2):PLAY"V2003":PLAY+Y$
430 IFPEEK(X)>148 AND PEEK(X)<175 THEN510
440 IFPEEK(X)>239 THEN POKEX+F,128:S=225:FORT=1TO4:S=390-S:SOUNDS,1:NEXT:RETURN
450 OF=X
460 POKEX,196
470 X=X-((X<&H420)*512)+((X)&H600)*480)-F
480 POKEOF,128
490 NEXT
500 POKEX+F,128:RETURN
510 R5=R5-(T2*F)+F:L5=L5-(T2*F)+F
520 POKEX+F,128:FORT=1TO6:FORT1=1TO3
530 C=(16*T1)+175:C$=STR$(T1*4):PLAY"V31":PLAYC$
540 IFX=R5 THEN X1=L:GOSUB600:NEXTT1:E=0:NEXT:HR=HR+1
550 IFX=L5 THEN X1=R:GOSUB600:NEXTT1:E=0:NEXTT:HL=HL+1
560 C=128:GOSUB600
570 IFX1=L THENX1=R:C=128:GOSUB600
580 IFX=L5 THEN X1=L:C=128:GOSUB600
590 Y=0:RETURN
600 POKEX1,C:POKEX1+1,C:POKEX1-31,C:POKEX1-32,C:RETURN
610 CLS:PRINT"*******************************";
620 PRINT"********* BATTLE TANKS *********";
630 PRINT"*******************************"
640 PRINT@160," USING YOUR JOYSTICK YOU HAVE "," TO DESTROY YOUR FRIENDS TANK","
 BEFORE HE GETS YOU.BE CAREFUL"," HE MAYBE HIDING AROUND THE "," CORNER......
650 PRINT@352," BE SURE TO CHOOSE YOUR FRIENDS"," WITH CARE!!!
660 PRINT@454,"PRESS SPACE TO PLAY
670 IFINKEY$<>" " THEN670
680 CLS
690 PRINT65,"PLEASE SELECT GAME:-
700 PRINT@168,"1----BATTLE FIELD
710 PRINT@205,"(IT'S A SHOOT OUT)
720 PRINT@264,"2----HIDE & SEEK
730 PRINT@301,"(MIND THE BARRIERS)
740 PRINT@360,"3----MINE FIELD
750 PRINT@397,"(BEWARE THE MINES)
760 PRINT@456,"4----FORCE FIELD
770 PRINT@493,"(HIDDEN BARRIERS)
780 G$=INKEY$:IFG$="" THEN780
790 IFVAL(G$)<1 ORVAL(G$)>4 THEN780
800 G=VAL(G$):CLS:GOTO830
810 CLS:GOTO830
820 PRINT@34,"NOW YOU KNOW WHO IS'NT YOUR "," FRIEND...WHY NOT TRY AND ","GET YO
UR OWN BACK!!!!
830 PRINT@165,"PLEASE SELECT:-
840 PRINT@232,"A----AMATEUR
850 PRINT@296,"P----PROFESSIONAL
860 PRINT@360,"E----TO END
870 A$=INKEY$:IFA$="" THEN870
880 IFA$="A" THENPOKE65494,0:GOTO920
890 IFA$="P" THENPOKE65495,0:GOTO920
900 IFA$="E" THENPOKE65494,0:FORT=0TO8:CLST:SOUND(T+1)*25,1:FORT1=1TO30:NEXTT1:N
EXT:SOUND25,4:END
910 GOTO870
920 CLS0
930 HL=0:HR=0
940 IFG=2 THEN100
950 IFG=3 THENGOSUB1000:GOTO110
960 IFG=4 THENGOSUB1010
```
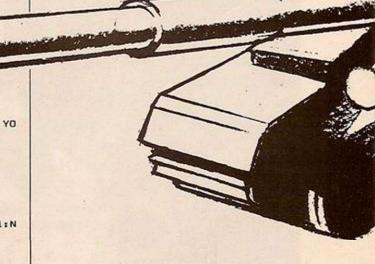
BAT

BATTLE TANKS is a Dragon 32 game for two players using the joystick controls. There is a choice of four options. First is an enclosed screen where each player blasts away at the other tank without any restrictions. However, the tanks have limited shell ranges. In the second version the players have barriers to hide behind to escape from their enemy. However, each can drive or fire off the edges of the picture, returning at the opposite side to catch the enemy unawares.

Minefields, the third option, is displayed on an apparently clear screen. Hidden within the picture are three randomly-positioned mines. If you ride over these mines you are blown up and lose a penalty point. The hidden mines are achieved by Poking on to the picture the value 240. This corresponds to the black in the orange colour set as opposed to the black value 128 normally displayed by the Dragon.

The final option, Force Field, employs the use of the value 240 and Pokes this on to the screen to form invisible walls to negotiate. The object of all the games is to score 10 hits to win. With the hidden mines and walls the player can fire their guns to determine the position of the obstructions as the shell automatically stops when they are detected.

To convert the program from joystick to keyboard operation add in the lines shown. The left-hand tank uses key 2 to go forward, Q-left, W-right, A-fire. The right-hand tank uses the minus sign to go forward, @ to go left, a left-pointing arrow to go right, and Enter to fire.

TLE TANKS

DRAGON

As the steel tide of war rips across the disputed zone, you get your free invitation to take part in the holocaust, courtesy of Vincent Crockett.
Can you and the crew of your main battle tank survive?

```
970 FORT=0TO31:POKE&H400+T,255:POKE&H5E0+T,255:NEXT
980 FORT=1TO14:POKE&H400+(T*32),255:POKE&H41F+(T*32),255:NEXT
990 GOTO110
1000 CLS0:FORT=0TO2:POKE&H421+RND(360),240:NEXT:RETURN
1010 FORT=0TO4:POKE&H487+T,240:POKE&H574+T,240:POKE&H50B+(T*32),240:POKE&H45A+(T
*32),240:NEXT
1020 RETURN
1030 A=42:SOUND4,1:GOSUB1050:A=159:GOSUB1050:A=102:SOUND15,1:GOSUB1050:A=175:SOU
ND50,1:GOSUB1050
1040 A=87:SOUND100,1:GOSUB1050:A=255:SOUND4,1:GOSUB1050:A=128:SOUND35,1:GOSUB105
0
1050 POKEX,A:POKEX+1,A:POKEX-1,A:POKEX+32,A:POKEX-32,A
1060 POKEX+2,A:POKEX-2,A:POKEX+64,A:POKEX-64,A
1070 POKEX+31,A:POKEX+33,A:POKEX-33,A:POKEX-31,A:RETURN
1080 PRINT@449,"AGAIN (Y/N) OR CHANGE GAME (C)";
1090 A$=INKEY$:IFA$="" THEN1090
1100 IFA$="Y" THENFORT=1473 TO1502:POKET,128:NEXT:GOTO920
1110 IFA$="N" THENA$="E":GOTO900
1120 IFA$="C" THEN680
1130 GOTO1090
1140 PRINT@449,"LOST ONE PENALTY POINT";
1150 FORT=1TO500:NEXT
1160 FORT=1473TO1495:POKET,128:NEXT:RETURN

Modifications for joystick control.

150 REM
170 IFPEEK(340)=254 THENX1=L:C=128:XL=1:GOSUB600
180 IFPEEK(343)=253 THEN X1=R:C=128:XR=1:GOSUB600
230 IFPEEK(R-M2)>239 ORPEEK(R-1-M2)>239 ORPEEK(R-31-(2*M2))>239 ORPEEK(R-32-(2*M
2))>239 THEN250
270 IFPEEK(338)=191 THENX=(R-(2*M2)):R5=X:F=M2:GOSUB420
280 IFPEEK(339)=251 THENX=(L-(2*M)):L5=X:F=M:GOSUB420
320 IFPEEK(339)=239 ORPEEK(345)=239 THEN AR=(AR+((PEEK(339)=239))*-1)+(PEEK(345)
=239)+((AR=0 ANDPEEK(345)=239)*-8)+((AR=7 ANDPEEK(339)=239)*8):M=A(AR,0):L1=A(AR
,1):L2=A(AR,2):L3=A(AR,3):L4=A(AR,4):GOSUB350
330 IFPEEK(338)=251 ORPEEK(343)=223 THENBR=(BR+((PEEK(338)=251)*-1)+(PEEK(343)=2
23))+((BR=0 ANDPEEK(343)=223)*-8)+((BR=7 ANDPEEK(338)=251)*8):M2=A(BR,0):R1=A(BR
,1):R2=A(BR,2):R3=A(BR,3):R4=A(BR,4):GOSUB360
```
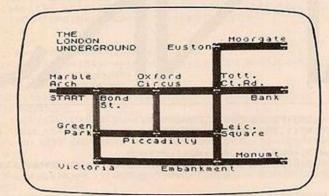
The B.B.C. Microcomputer Model B

available today, but designed for the future...

Are you looking for a home computer that is **flexible** - that can cover education, entertainment or business; **powerful** - that offers 32K of memory; and is above all **great value**?

Then look no further! Come and see us at **Microstyle** and we'll show you the **B.B.C. Micro'**.

A home computer system designed to grow hand-in-hand with the abilities of it's user, and that is capable of keeping pace with advances in modern technology.

We stock a wide range of exciting games and systems packages, quality add-on facilities and a comprehensive selection of accessories designed to expand your B.B.C. system.

FROG ON T[

The truckers on this four-lane freeway are too busy chomping chocolate bars and watching out for sports cars to worry about a jaywalking frog: so look both ways in Desmond Phillips' Frogrun.

```
1 REM FROGRUN
2 REM (C) D.PHILLIPS
10 POKE36879,254:PRINT"         FROGRUN        ":PRINT"CONTROLS":PRINT"
U"
20 PRINT"  !":PRINT" H+J":PRINT"  !":PRINT" N"
30 PRINT"N DODGE THE TRUCKS AND  CATCH THE LOGS!"
40 PRINT"N YOU HAVE 3 LIVES AND  30 SECONDS TO GUIDE    YOUR FROG HOME"
50 PRINT"N ONCE ALL 3 FROGS ARE  HOME YOUR TIME LIMIT  WILL DECREASE":GOSUB650
CLR
60 SC=0:L=3:S1=36876:F1=32:F2=32:F3=32:TX=1800:FT=0
70 OP=32:OC=1:Y=10:X=15:TI$="000000":POKE36878,15
80 PRINT"X////////        ":PRINT"N////////        ////"
90 PRINT"X      ////      //////        ////"
100 PRINT"X//        ////////////////////"
110 PRINT"I (%)      (%)     (%) (%)":PRINT"%% $%&   $%& $%&"
120 PRINT"XK(%)      (%)      (%) (%)":PRINT"X  $%&    $%&    $%&XX"
130 POKE7685,F1:POKE7694,F2:POKE7698,F3:POKE38405,5:POKE38414,5:POKE38418,5
140 DY=0:DX=0:A=PEEK(203):IFA=51THENDY=-1:POKES1,180
150 IFA=28THENDY=1:POKES1,190
160 IFA=43THENDX=-1:POKES1,200
170 IFA=20THENDX=1:POKES1,210
180 POKES1,0
190 IFX+DX<10RX+DX>20THEN420
200 IFY+DY>10THEN300
210 P=PEEK(7680+X+DX+44*(Y+DY))
220 IFY>1ANDY<6ANDP=32THEN420
230 IFY>6ANDP<>32ANDP<>35THEN420
240 IFY+DY=0ANDP<>32THEN420
250 IFY+DY=0ANDP=32THEN520
260 IFDY=0ANDDX=0THEN290
270 POKE7680+X+44*Y,OP:POKE38400+X+44*Y,OC
280 OP=PEEK(7680+X+DX+44*(Y+DY)):OC=PEEK(38400+X+DX+44*(Y+DY)):SC=SC+10
290 X=X+DX:Y=Y+DY:POKE7680+X+44*Y,35:POKE38400+X+44*Y,5
300 IFY=10RY=3THENX=X-1
310 IFY=20RY=4THENX=X+1
320 IFY>5ANDY<10THENPOKE7680+X+44*Y,32
330 POKE2,30:POKE1,44:SYS820:POKE1,88:SYS866:POKE1,132:SYS820:POKE1,176:SYS866
340 POKE2,31:POKE1,8:SYS820:POKE1,52:SYS866:POKE1,96:SYS820:POKE1,140:SYS866
350 P=PEEK(7680+X+44*Y):IFP<>32ANDP<>35THEN420
360 IFY>5ANDY<10THENPOKE7680+X+44*Y,35:POKE38400+X+44*Y,5
370 IFSC>HITHENHI=SC
380 PRINT"T"TI$"  L"L
390 PRINT"S "SC"J":PRINT"IIIIIIIIIIH "HI"I"
400 IFTI>TXTHEN420
410 GOTO140
420 POKE7680+X+44*Y,OP:POKE38400+X+44*Y,OC
430 POKE7680+DX+X+44*(Y+DY),35:POKE38400+X+DX+44*(Y+DY),2
440 FORT=15TO0STEP-.3:POKES1-1,128+2*T:POKE36878,T:NEXT:POKE36875,0
450 L=L-1:IFL=0THEN470
460 FORT=1TO1000:NEXT:GOTO70
470 PRINT"XXXX ANOTHER GAME [Y/N]?"
480 POKE198,0
490 GETA$:IFA$="Y"THEN60
500 IFA$<>"N"THEN490
510 POKE2,0:SYS2
520 POKE7680+X+44*Y,OP:POKE38400+X+44*Y,OC
530 POKE7680+X+DX+44*(Y+DY),35:POKE38400+X+DX+44*(Y+DY),5
540 FORT=1TO30:FORG=200TO210:POKES1,G+T:NEXT:NEXT:POKES1,0
550 FORT=1TO1000:NEXT:SC=SC+500
560 IFX=5THENF1=35
570 IFX=14THENF2=35
580 IFX=13THENF3=35
590 FT=FT+1:IFFT=3THENFT=0:F1=32:F2=32:F3=32:TX=TX-300:SC=SC+1000
600 GOTO70
610 DATA 165,1,133,251,165,2,24,105,120,133,252,162,20,160,0,177,1,72,200,177,1
136,145
620 DATA 1,200,104,145,1,136,177,251,72,200,177,251,136,145,251,104,200,145,251
202,16
630 DATA 226,96,165,1,133,251,165,2,24,105,120,133,252,162,20,160,20,177,1,72,2
00,177,1
640 DATA 136,145,1,200,104,145,1,136,177,251,72,200,177,251,136,145,251,104,200
,145,251
650 DATA136,136,202,16,224,96:FORT=820TO913:READK:POKET,K:X=X+K:NEXT:IFX<>12468
THENSTOP
660 DATA 153,153,126,24,219,189,153,129,7,63,127,127,127,127,63,7,0,255,255,255
,255
670 DATA 255,255,0,112,254,241,241,241,241,254,112,255,255,255,255,255,255,255,
255
680 DATA 7,63,71,71,71,71,63,7,112,254,255,255,255,255,254,112,7
690 POKE56,28:POKE52,28:POKE51,0:POKE55,0:FORT=0TO511:POKE7168+T,PEEK(32768+T):
NEXT
700 FORG=7448TO7503:READK:POKEG,K:NEXT:POKE36869,255:RETURN
```

VIC-20
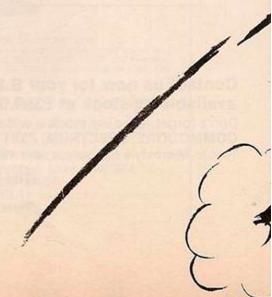
THE GAME Frogrun is designed for the unexpanded Vic-20 and uses machine code and defined graphics.
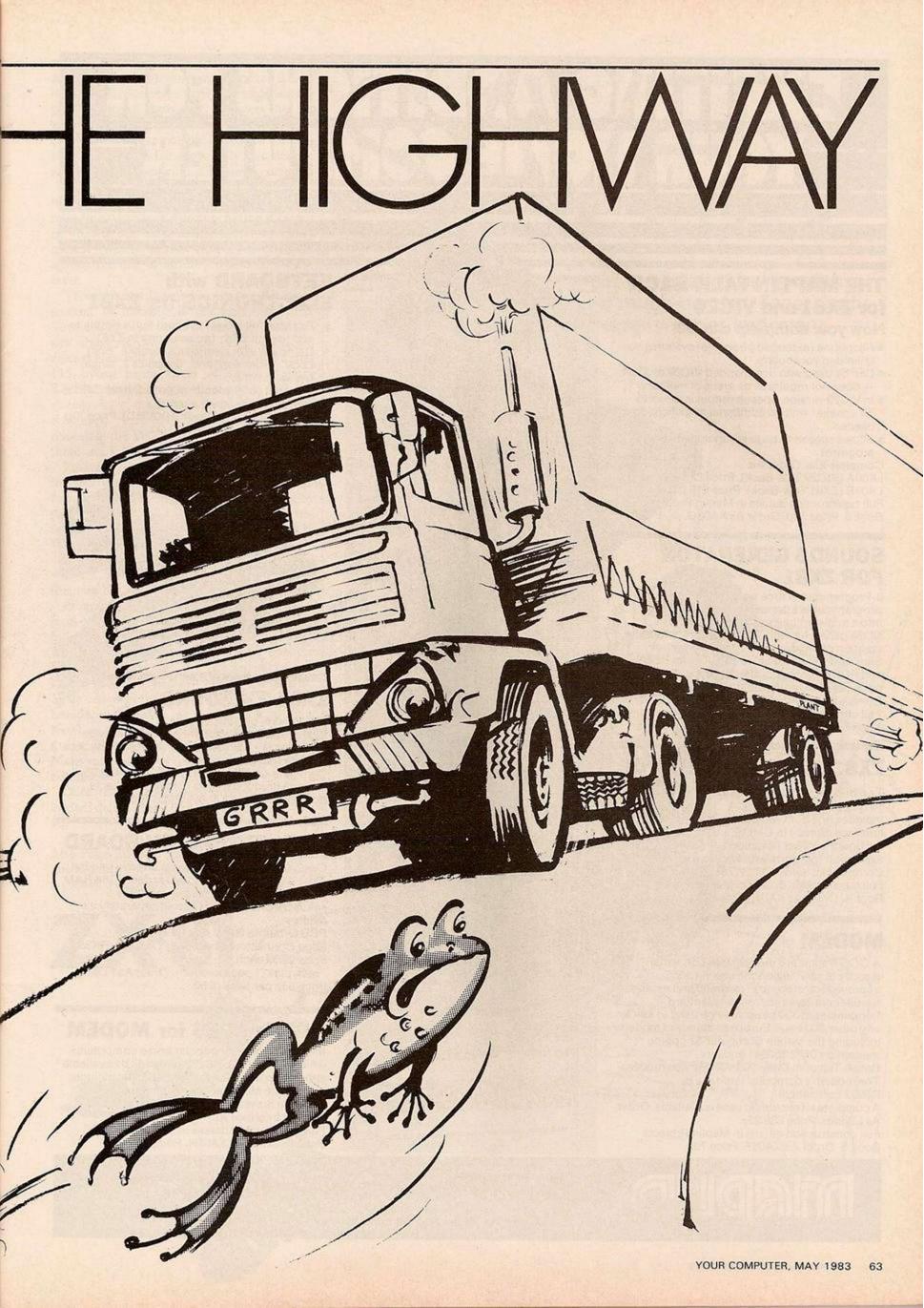
You have to guide an intrepid frog across four lanes of busy highway and safely on to the pavement. He then has to hop between the logs and make it to his home in a gap in the bank. There is a 30-second time limit — this goes down by five when three frogs have made home. You are not allowed near the edge of the screen or on the river bank.

The contest finishes when three frogs have been flattened. A main loop is incorporated and various subroutines. The scrolling is done by two machine-code routines in the cassette buffer at locations 820 and 866. These scroll right and left for one line whose starting address is at location 1 and 2. Unfortunately, the machine code and defined graphics take up a lot of memory as they have to be loaded from Basic.

10-50	Title screen
60-70	Initialise variables
80-130	Print game screen
140-180	Player input
190-260	Check move
270-290	Move frog
300-360	Scroll logs and trucks
370-410	Print Time, Score
420-510	Splat!
520-600	Frog Home!
600-650	Machine-code data
650-700	Defined graphics.

To stop painful crashes, there is an error trap in 650 that checks the sum total of the machine code data and Stops it if there is a discrepancy. The controls U, H, J and N are read by Peeking location 203 and can be changed be reference to the table on page 179 of the Reference Guide. My own personal high score is 13,410.

HE HIGHWAY

EXCITING ADDITIONS FOR YOUR HOME COMPUTER

THE MAPLIN TALK-BACK
for ZX81 and VIC20
Now your computer can talk

★ Allophone (extended phoneme) system gives unlimited vocabulary.
★ Can be used with unexpanded VIC20 or ZX81 — does not require large areas of memory.
★ In VIC20 version, speech output is direct to TV speaker with no additional amplification needed.
★ Allows speech to be easily included in programs.

Complete kits. Order As:
LK00A (VIC20 Talk-Back). Price £24.95.
LK01B (ZX81 Talk-Back). Price £19.95.
Full construction details in Maplin Projects Book 6. Price 70p. Order As XA06G.

SOUNDS GENERATOR FOR ZX81

3-Programmable tone generators with programmable attenuators turns your ZX81 into a mini-synthesiser.
Noise generator with 3 pitch levels enables the creation of most special effect sounds.
Single address access via BASIC with PEEK and POKE. Connects directly to expansion port socket or motherboard.
Complete kit. Order As LW96E. Price £10.95.
Full construction details in Maplin Projects Book 5. Order As XA05F. Price 70p.

ZX81 INPUT/OUTPUT PORT

Two bi-directional ports for a total of 16 input or 16 output lines.
Includes one buffered output which can interface directly to CMOS.
On board address selection allows for expansion to 6 ports with two boards.
Complete kit. Order As LW76H. Price £9.25.
Full construction details in Maplin Projects Book 4. Order As XA04E. Price 70p.

MODEM

A CCITT standard modem that connects directly to your telephone line via a B.T. approved transformer. Transmits and receives simultaneously on European standard frequencies at 300 baud. May be used to talk to any other 300 baud European standard modem including the Maplin Computer Shopping modem on 0702 552941 and any British Telecom Datel 200/300 Service modem. The modem's computer interface is RS232 compatible.
A complete kit excluding case is available. Order As LW99H. Price £39.95.
Full construction details in Maplin Projects Book 5. Order As XA05F. Price 70p.

KEYBOARD with ELECTRONICS for ZX81

★ Full size, full travel keyboard that's simple to add to your ZX81 (no soldering in ZX81).
★ Complete with electronics to make "Shift Lock", "Function" and "Graphics 2" single key selections.
★ Powered (with adaptor supplied) from ZX81's own standard power supply.

Full details in Project Book 3 (XA03D). Price 70p.
Complete kit (excl. case) £21.90.
Order As LW72P.
Case £4.95. Order As XG17T.
Ready built-in case £29.95.
Order As XG22Y.

ZX81 TV SOUND

ZX81 sound on your TV set.
Video reversing switch for normal or inverse video display. Can be used with the Talk-Back or Sounds Generator kits.
Complete kit. Order As LK02C. Price £19.95.
Full construction details in Maplin Projects Book 6. Order As XA06G. Price 70p.

ZX81 EXTENSION BOARD

Our ZX81 Extendiboard plugs directly into the ZX81 expansion port and will accept a 16K RAM pack and three other plug-in modules simultaneously. Parts are sold separately as follows:
PCB Order As GB08J. Price £2.32.
Edge Connectors (4 needed) Order As RK35Q. Price £2.39 each.
Track pins (1 pack needed). Order As FL82D. Price 85p per pack of 50.

INTERFACES for MODEM

Interfaces for most popular home computers which do not have RS232 ports will be available shortly. An interface for the ZX81 that includes a complete Machine Code Communications Program is available. Order As LK08J (ZX81 Modem Interface Kit). Price £24.95.
Full construction details in Maplin Projects Book 7. Order As XA07H. Price 70p.

MAPLIN

SAVED!

Andy Scott helps you attain the goal of better saving on the ZX-81 by providing a much-needed Tape Verify routine — amount of memory is not important.

AN UNFORTUNATE shortcoming of the ZX-81 is the absence of a tape verify. Considering the poor quality of many tape recorders that get used with ZX-81s, and the widespread use of cassettes that have not been certified for data storage, there is a fair probability that, once in a while, a program will not be saved correctly.

Nothing is more frustrating than spending hours typing in a program, saving it and reloading it at a later date only to find it cannot be loaded due to a saving error. A Verify routine which checks the saved program against the one in memory would be a great asset.

This program is suitable for any ZX-81 — amount of memory is not important. The Verification program is in machine code and sits above RAMtop, thus being protected against New and Load. The program occupies 116 bytes, hence users of small memory ZX-81s should bear this in mind. If users already have machine-code programs above RAMtop, these will not be destroyed or relocated: the Verify program will sit between these and RAMtop.

Program 1 resets RAMtop to accommodate the routine. It looks at the existing RAMtop, reduces it by 116 bytes, then destroys itself by performing New. Because it is self-destructive, remember to Save it on tape before running it! While running, it comes up with:

        TO VERIFY RAND USR X

where x is a number shown. Make a note of this number — it will be used later.

Program 2 is the verify routine. It sits in a Rem statement in line 1. After typing Rem, a follow it with 116 characters — I usually use 0. To load the machine-code program into the Rem, Run program 2. The machine-code loader used has become pretty standard in *Your Computer*. Enter the code in hexadecimal, either as a single byte plus Newline, for example, 76 N/L, or as a block at a time for example 7676CD230F plus N/L. Make sure each byte consists of two digits for example 0F and not F.

Once the machine-code program has been loaded, delete lines 10 to 90. Then put in line 10:

        10 RAND USR 16632

Also Poke 16510,0. This will change the Rem line number to 0 — a safer approach since line 0 cannot be deleted. Now Save program 2 on

ZX-81

tape. The two Halt instructions at the beginning of the machine-code program are there to prevent the contents of the Rem statement from being listed. This is preferable for two reasons — one to prevent an awful mess from being listed and, secondly, if the machine-code program is longer than can be listed on to the screen, the ZX-81 will effectively crash.

Now Run the program. It will relocate the routine to above RAMtop, changing three absolute Calls at the same time to the correct address. It will then perform New — thus destroying the Basic program.

Now to use Verify. First put in a Basic program and Save it. Rewind the tape back to the start of the program just saved, then Rand Usr x as a direct command — where x is that number you wrote down from program 1 — remember? Now press Play on the tape recorder. Three types of report code can be shown:

0/0 program verified and is OK
D/0 Break key pressed during Verify
E/0 Error has been found on tape.

D/0 is similar to when Break is pressed during normal Load and Save. Note the use of E/0 — Sinclair does not use this report code. If an error has been found, your original program has not been destroyed — try listing it! Just re-Save the program and do Verify again.

The Verify routine does not check if system variables have been saved satisfactorily — during Verification many would be different to when saved. In between saving and verifying, do not change any user variables or define new ones — they would be different to those stored on tape and an error would be shown. The Verify program can be used within a Basic program also, for example:

```
100 SAVE "EXAMPLE"
110 PRINT "REWIND TAPE TO START OF
    EXAMPLE"
120 PRINT "THEN PRESS PLAY ON TAPE
    RECORDER"
130 PRINT "PRESS G KEY WHEN YOU HAVE
    DONE THIS"
140 IF INKEY$ <> "G" THEN GOTO 140
150 RAND USR X
```

Where X is the start of the Verify program — again the one you wrote down. ∎

```
Program 1.
10 LET X=PEEK 16388+256*PEEK 16389
20 LET X=X-116
30 LET T=X/256
40 LET H=INT T
50 LET L=(T-H)*256
60 POKE 16388,L
70 POKE 16389,H
80 PRINT "TO VERIFY USE  RAND USR ";X
90 PRINT "   PRESS ANY KEY TO CONTINUE"
100 IF INKEY$="" THEN GOTO 100
110 NEW
```

```
Program 2.
1 REM (fill this with 116 characters)
10 LET X=16514
20 LET A$=""
30 IF A$="" THEN INPUT A$
40 IF A$="S" THEN STOP
50 POKE X,16*CODE A$+CODE A$(2)-476
60 PRINT AT 11,7;X;" ",A$(1 TO 2)
70 LET X=X+1
80 LET A$=A$(3 TO)
90 GOTO 30
RUN (in FAST)
```

The Verify routine in machine code.

```
4082        76          HALT                        ;To prevent screen listing
4083        76          HALT
4084        CD230F      CALL  0F23,FAST
4087        37          SCF
4088        CB12        RL    D
408A        CB0A        RRC   D
408C  NEXT PROG CD9140  CALL  4091,INBYTE            ;Start tape listen
408F        18FB        JR    408C,NEXT PROG
4091  IN BYTE   0E01    LD    C,01
4093  NEXT BIT  0600    LD    B,00
4095  BREAK1    3E7F    LD    A,7F
4097        DBFE        IN    A,(+FE)                ;Test for BREAK key
4099        D3FF        OUT   (+FF),A                ;Echo to screen
409B        1F          RRA
409C        3805        JR    C,40A3,CONT1           ;JR if BREAK not pressed
409E  BREAK2    CD2B0F  CALL  0F2B,SLOW
40A1        CF          RST   08H                    ; > REPORT CODE D
40A2        0C          DEFB  0C                     ; > BREAK key pressed
40A3  CONT1     17      RLA
40A4        17          RLA
40A5        3834        JR    C,40DB,GET BYTE        JR Build up byte in C Reg.
40A7        10EC        DJNZ  4095,BREAK1
40A9        F1          POP   AF
40AA  PROG NAME CD9140  CALL  4091,IN BYTE
40AD        79          LD    A,C                    ;Load built up byte into A
40AE        17          RLA                          ;Bit 7 of byte into Carry
40AF        30F9        JR    NC,40AA,PROG NAME      ;JR if not last byte of name
40B1        210940      LD    HL,4009,VERSN          ;Start loading from 4009
40B4  VERIFY    CD9140  CALL  4091,IN BYTE           ;Byte from tape into C reg.
40B7        117940      LD    DE,4079                ;Start of Basic prog. into DE
40BA        A7          AND   A                      ;Clear carry
40BB        E5          PUSH  HL                     ;Save address count
40BC        ED52        SBC   HL,DE                  ;Are we up to Basic prog?
40BE        E1          POP   HL                     ;Retrieve address count
40BF        3804        JR    C,40C5,CONT2           ;And if not,do not verify
40C1        79          LD    A,C                    ;Byte from tape into A
40C2        BE          CP    (HL)                   ;Do Byte VERIFY
40C3        2011        JR    NZ,40D6,ERROR          ;JR ERROR if verify false
40C5  CONT2     23      INC   HL                     ;HL points to next address
40C6        ED5B1440    LD    DE,(E-LINE)
40CA        A7          AND   A                      ;Clear carry
40CB        E5          PUSH  HL                     ;Save address count
40CC        ED52        SBC   HL,DE                  ;Are we up to E-LINE?
40CE        E1          POP   HL                     ;Retrieve address count
40CF        20E3        JR    NZ,40B4,VERIFY         ;And if not cont. to verify
40D1  OK        CD2B0F  CALL  0F2B,SLOW
40D4        CF          RST   08H                    ;>No errors -
40D5        FF          DEFB  FF                     ;>REPORT CODE 0
40D6  ERROR     CD2B0F  CALL  0F2B,SLOW
40D9        CF          RST   08H                    ;>ERROR -
40DA        0D          DEFB  0D                     ;>REPORT CODE E
40DB  GET BYTE  D5      PUSH  DE
40DC        1E94        LD    E,94                    ;  Timing
40DE  TRAILER   061A    LD    B,1A                    ;  Loops
40E0  COUNTER   1D      DEC   E
40E1        DBFE        IN    A,(+FE)                 ;Pick up tape signal
40E3        17          RLA
40E4        CB7B        BIT   7,E
40E6        7B          LD    A,E                     ;  Build up
40E7        38F5        JR    C,40DE,TRAILER          ;  Byte in
40E9        10F5        DJNZ  40E0,COUNTER            ;  C Register
40EB        D1          POP   DE
40EC        2004        JR    NZ,40F2,BIT DONE
40EE        FE56        CP    56
40F0        30A1        JR    NC,4093,NEXT BIT
40F2  BIT DONE  3F      CCF
40F3        CB11        RL    C
40F5        309C        JR    NC,4093,NEXT BIT
40F7        C9          RET                           ;RETURN
40F8  BLOCKMOVE 2A0440  LD    HL,(RAMTOP)            ;Put address of RAMTOP in HL
40FB        010D00      LD    BC,000D
40FE        09          ADD   HL,BC                  ;address for INBYTE in HL
40FF        228D40      LD    (408D),HL
4102        22AB40      LD    (40AB),HL
4105        22B540      LD    (40B5),HL
4108        218440      LD    HL,4084                ; >Transfer programme
410B        ED5B0440    LD    DE,(RAMTOP)            ; > (4084 to 40F7)
410F        017400      LD    BC,0074                ; >to above RAMTOP
4112      . EDB0        LDIR
4114        CDC303      CALL  03C3,NEW
```

```
Users of old Sinclair ROMs note the following changes:
    4084-4086    CD200F
    409E-40A0    CD280F
    40D1-40D3    CD280F
    40D6-40D8    CD280F
Or alternatively, load the machine code programme as described above,
then POKE 16517,32; POKE 16535,40; POKE 16594,40; POKE 16599,40
```

```
   1 REM SATURN LANDER
   2 REM © G. OWEN,MARCH 1983
   3 GO TO 20
  10 PLOT SX,SY: PLOT SX,SY+1: D
RAW 3,0: PLOT SX+3,SY: PLOT SX+1
,SY+2: PLOT SX+2,SY+2
  11 RETURN
  20 PAPER 0: BORDER 0: INK 5: C
LS : OVER 1
  21 PLOT 0,0: DRAW 10,20: DRAW
9,-10: DRAW 8,6: DRAW 14,-10: DR
AW 6,4
  22 DRAW 4,-4: DRAW 8,6: DRAW 1
4,-10: DRAW 16,0
  24 DRAW 0,1: DRAW -16,0: DRAW
OVER 0;16,-1
  26 DRAW 14,6: DRAW 4,-4: DRAW
18,8: DRAW 6,-2: DRAW 3,4
  28 DRAW 10,-10: DRAW 10,6: DRA
W 8,12: DRAW 8,4
  30 DRAW 16,-10: DRAW 24,14: DR
AW 8,-10: DRAW 4,2: DRAW 8,-10
  32 DRAW 10,-4
  35 RANDOMIZE : FOR S=1 TO RND*
100: LET L=USR 32300: NEXT S
  40 LET R$="AB  CD  AB  CD AB
CD  AB AD CB "
  45 PRINT AT 12,0;R$; AT 15,0;R$
: LET R$=R$(15 TO )+R$( TO 14):
PRINT AT 17,0;R$
  50 LET R$="EFG   KLM   KLM   EF
G  EFG   KLM  HIJ  NOP   NOP   HI
J  HIJ  NOP  "
  55 PRINT INK 5;AT 4,0;R$;AT 8,
0;R$
  60 LET SX=RND*100+75: LET SY=1
85
  61 LET HV=0: LET VV=0
  62 LET S=0
  64 GO SUB 10
  65 IF POINT (SX,SY-1)+POINT (S
X+1,SY-1)+POINT (SX+2,SY-1)=3 TH
EN GO TO 2000
  66 LET OSX=SX: LET OSY=SY
  67 LET SX=SX+HV
  68 LET SY=SY-VV: IF SY>165 THE
N LET VV=0
  69 LET NSX=SX: LET NSY=SY: LET
 SX=OSX: LET SY=OSY
  70 GO SUB 10
  71 LET SX=NSX: LET SY=NSY
  72 LET S=S+1-(INKEY$<>"")
  75 LET L=USR 32200
  76 LET HV=HV+.25*((INKEY$="2"
AND SX<250 AND HV<=3)-(INKEY$="1
" AND SX>2 AND HV>=-3))
  77 LET VV=VV+.25-.5*(INKEY$="0
")
  78 INPUT "": PRINT #1;"VERT.VE
L=";INT (VV*100)/100,"HOR.VEL=";
INT (HV*100)/100
  80 IF POINT (SX,SY)+POINT (SX,
SY+1)+POINT (SX+1,SY+1)+POINT (S
X+1,SY+2)+POINT (SX+2,SY+1)+POIN
T (SX+2,SY+2)+POINT (SX+3,SY)+PO
INT (SX+3,SY+1)<>0 THEN GO TO 10
00
 200 GO TO 64
1000 FOR F=0 TO 30: OVER (F/2=IN
T (F/2)): INK 2: PLOT SX,SY: DRA
W 3,2
1010 PLOT SX,SY+2: DRAW 3,-2
1020 OUT 254,55*(F/2=INT (F/2))
1030 NEXT F
1035 INK 7
1040 OUT 254,0
1050 PRINT OVER 0;AT 10,11; FLAS
H 1;"GAME OVER"
1055 PRINT TAB 11;"SCORE=";S
1060 STOP
2000 IF ABS HV>.25 OR SY>5 THEN
GO TO 1000
2010 FOR X=10 TO 50 STEP 5: BEEP
 .1,X: NEXT X
2015 INK 7
2020 PRINT AT 10,12; FLASH 1;"WE
LL DONE"
2030 PRINT TAB 11;"SCORE=";200-S
8998 STOP
8999 PRINT "PLEASE WAIT WHILE DA
TA IS POKED"
9000 RESTORE 9000: FOR U=0 TO 12
7
9010 READ DATUM: POKE USR "A"+U,
DATUM: NEXT U
9020 DATA 0,15,31,31,63,63,7,0,0
```

```
,192,224,248,252,252,192,0
9030 DATA 0,1,15,31,127,63,31,0,
0,224,252,254,254,248,240,0
9040 DATA 0,0,0,3,15,31,63,61,0,
0,0,248,254,255,255,255,0,0,0,0,
0,0,128,192
9050 DATA 126,127,127,127,127,63
,15,0,247,15,255,255,255,255,255
,255,224,240,BIN 10110000,BIN 10
111000,BIN 01111000,DATUM,240,0
9060 DATA 0,0,0,0,0,1,3,15,0,0,0
,0,127,255,255,255,0,0,0,0,128,2
24,240,248
9070 DATA BIN 00110111,111,111,B
IN 00110111,31,3,0,0,251,247,239
,223,255,255,0,0,248,248,240,240
,128,0,0,0
9500 RESTORE 9500: READ clr: CLE
AR clr
9505 READ ADD
9510 LET A$=""
9520 IF A$="" THEN READ A$
9525 IF A$="N" THEN GO TO 9505
9526 IF A$="S" THEN STOP
9540 IF CODE A$(1)>64 THEN LET A
$(1)=CHR$ (CODE A$(1)-7)
9550 IF CODE A$(2)>64 THEN LET A
$(2)=CHR$ (CODE A$(2)-7)
9660 POKE ADD,16*CODE A$+CODE A$
(2)-816
9570 LET ADD=ADD+1
9580 LET A$=A$(3 TO )
9590 GO TO 9520
9600 DATA 31999
9610 DATA 32000,"C5D5E5F52B11200
00808C5237ECB172BF519F10620CB152
B10FB24C110ECF1E1D1C1C9N"
9620 DATA 32100,"C5D5E5F51120000
0808C52B197ECB1FF5A7ED5223F10620C
B1E2310FBA7ED5224C110E6F1E1D1C1C
9N"
9630 DATA 32200,"218040CD007D21A
040CD007D210048CD647D212048CD647
D218048CD007D21E048CD647D212050C
D007DC9N"
9640 DATA 32300,"216050CD647D218
050CD647D21A050CD647DC9N"
9700 DATA 0,"S"
```

SATURN LANDER

Landing on the surface of the planet Saturn from an equatorial orbit is not an easy number. Garry Owen's 16K Spectrum program will help you run rings round space pilots who boast about working in the asteriod belt.

YOU ARE IN command of the first manned mission to Saturn. Your object is to land on the planet's surface but to get there you have to navigate through the rocks which make up Saturn's rings whilst compensating for the strong gravity.

The program, for a 16K Spectrum, is mainly Basic but four machine-code routines are used. The routine at 32000-32024 scrolls one line right to left by one pixel and the routine at 32100-32139 scrolls one line left to right by one pixel.

Each of these needs the address of the first byte of the top row of the line to be scrolled supplied to it in the hl register to work.

The routine at 32200-32242 calls the two scroll routines and provides the correct addresses. This moves the rocks. The routine at 32300-32318 positions the landscape so that the landing pad is a random number of pixels from the edge of the screen.

The data for the machine-code routines is held in Data statements, in hexadecimal, in lines 9610-9640 and Poked into memory by lines 9500-9590 after RAMtop has been lowered to 31999 — lines 9500 and 9600.

The data for the user-defined graphics — lines 9020-9070 — is Poked into place by lines 9000 and 9010. Note that the capital letters in lines 40 and 50 are the graphics characters on those keys.

Type in the program as shown and before attempting to Run it, Save it so that it auto-runs from line 8999 on Loading, that is

SAVE"SATURN"LINE 8999

This ensures that all the data is set up before the program is Run. Verify it, New the program and reload. You should get the message

PLEASE WAIT WHILE DATA IS POKED

The program is now ready to run.
The controls are:

    1 to accelerate to the left
    2 to accelerate to the right
    0 to accelerate upwards

Note that if you are already moving in the opposite direction to that which you want to go, you will slow down first. Watch the velocity readings at the bottom of the screen.

The remaining program structure is as follows:

10-11	draw lander
20-35	draw landscape and move it
40-55	prints rocks on screen
60-62	initialise variables
64-200	main program
1000-1060	crash routine
2000-2030	successful landing routine
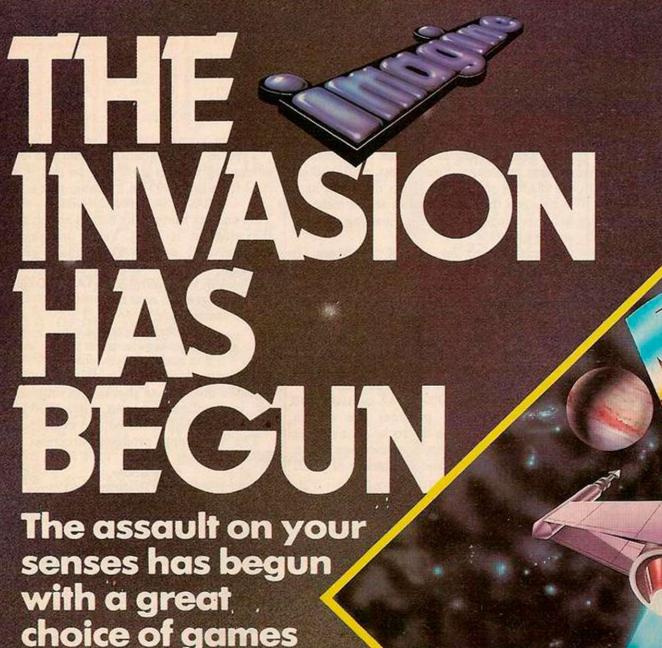
The variables used are SX and SY which give the co-ordinates of the lander, and HV and VV — horizontal and vertical velocities. ∎
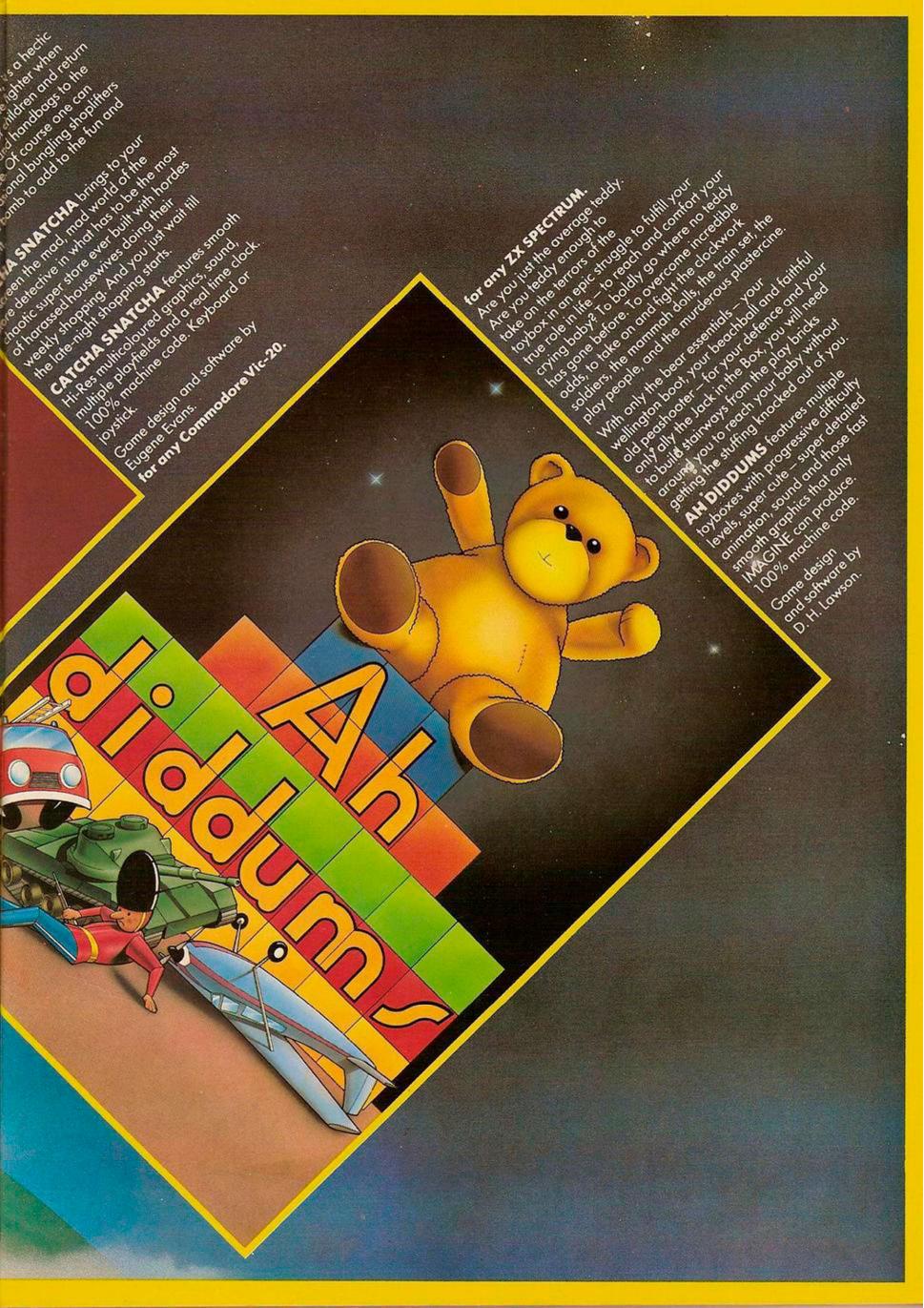
ZX-81

RUN SILENT, RUN DEEP

Stand by for action as your hydrofoil destroyer remorsely tracks a submerged sonar target. Rod Hopkins takes you on a voyage — hopefully not to the bottom of the sea.
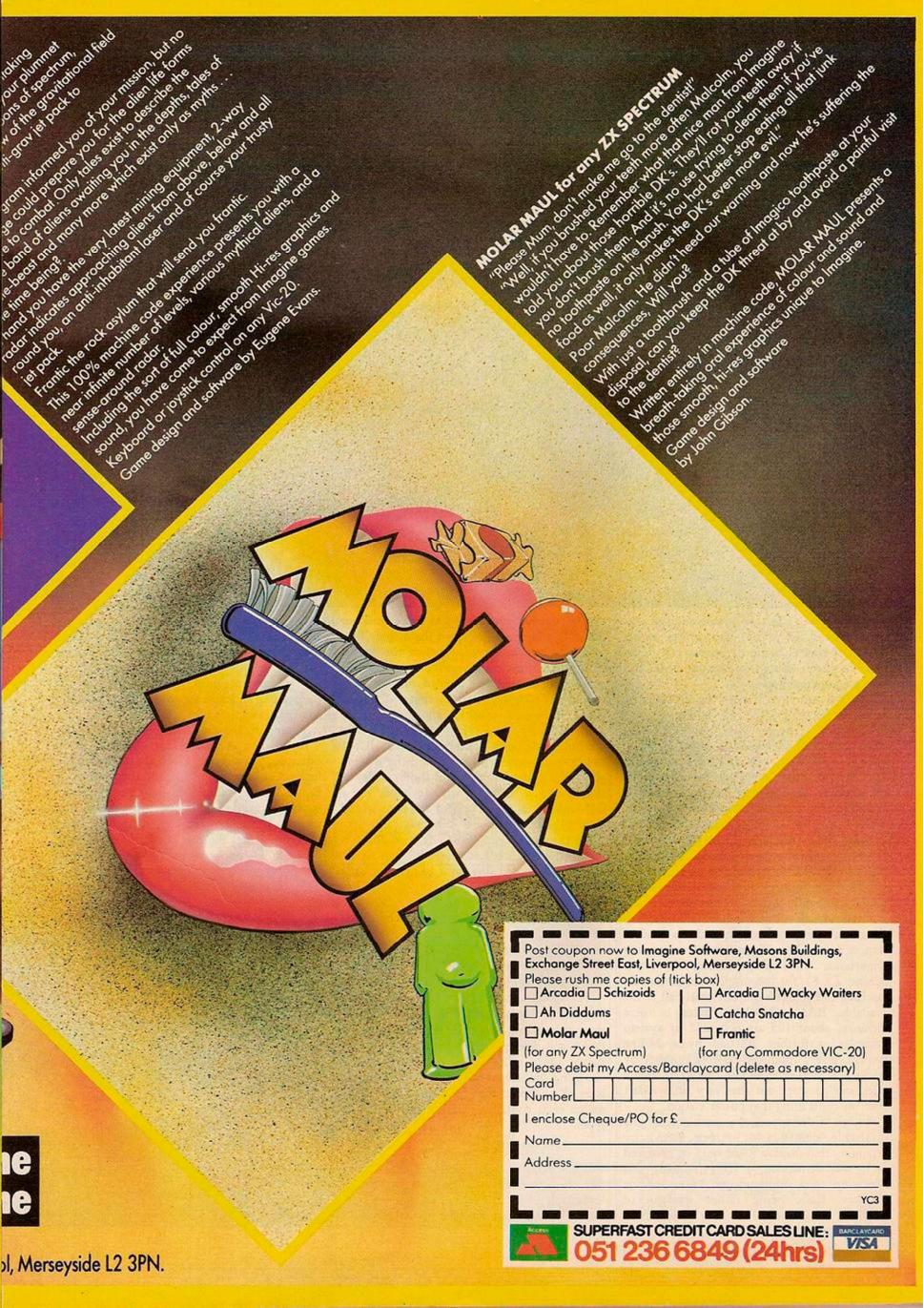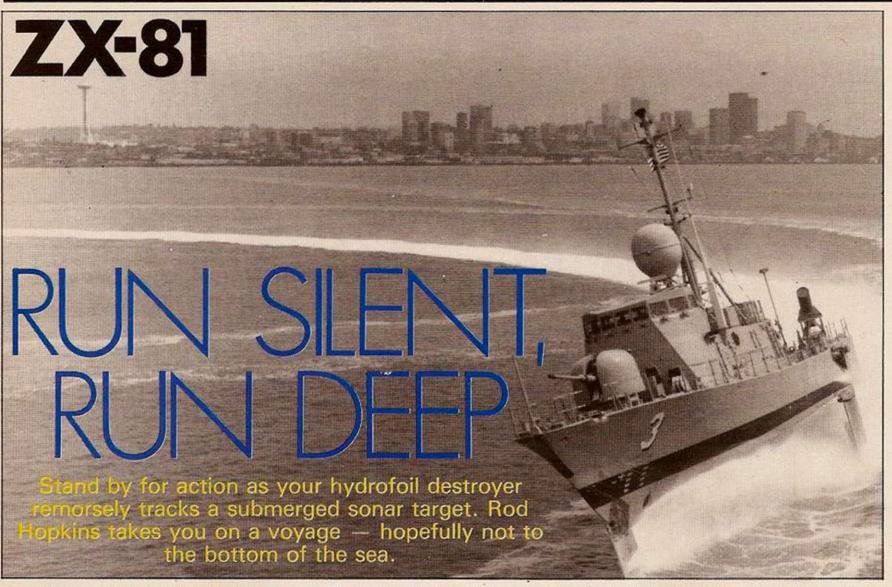
A DESTROYER LURKS on a storm-tossed sea, lying in wait for passing submarines — the object, of course, being to destroy one. This game has been written in machine code, and uses imaginative display.

The first and most obvious benefit accruing from the use of machine code is that of speed of execution. Basic is a general-purpose, high-level language designed to cope with an infinite variety of programming needs. A machine-code programmer, on the other hand, can confine his attention to the particular needs of the moment. The result is that a lot of time-consuming classification of tasks, of fetching and carrying can be avoided. A Basic program is ultimately performed as a machine-code sequence of course, but an awful amount of sorting and organising must be carried out by the interpreter during its execution.

A second, and directly-related advantage to be gained from writing in code is that of compactness. This factor is of overriding importance when programming the unexpanded ZX-81, for the simple reason that the machine can arrogate to itself nearly all of the available RAM. Of 1024 bytes of user memory, about 125 are occupied by system variables; a further 700-odd by a 22-line display, and a smallish, variable number by stacks and marker bytes. You will not need a calculator to work out that very little is left over for controlling that power station.

Sinclair sidestepped the problem of the insatiable demands of screen RAM by arranging that display RAM only be allocated as it was demanded, a CLS reducing this allocation to a minimum of 25 end-of-line markers. Basic being rather wasteful of memory, programmers in that language for the 1K machine are forced to make constant use of this facility to contract the display file.

The technique is this: to clear the screen each program cycle and reprint the display, in order that unused but opened display lines should not accumulate an overhead of wasted RAM. Economy results, but unfortunately with a fairly paralysing strobe-effect. In machine code, a sufficiency of display RAM can be permanently opened, and the display given a feel of continuity and fluidity. Naturally, if enough memory were to hand, and a fast interpreter, then the pros and cons would need to be reargued.

For the current program, a decimal dump is provided in figure 1, disassembled in figure 2. It will be noted that some use of Sinclair's ROM has been made.

The first address shown in the dump is 16514, indicating that the code is to be stored in a Rem statement occupying the first Basic line of the program. Therefore, type:

    1 REM 110 CHARACTERS . . .
whichever you like.

Having done this, type

    PRINT PEEK 16511

the result should be 112. If it is not, then edit line 1 and delete or add characters as appropriate. Once line 1 is correct, edit it twice, changing the line number to 2 and 3 respectively. You should now have three Rem statements each containing 110 characters. Now type:

    POKE 16511,88
    POKE 16512,1

and in effect, a single Rem statement having 342 characters remains. The stage is now set for entering the machine code. To prevent an unbreakable listing loop occurring after the

code has been entered, the two first characters Poked will be Newlines, or end-of-line markers. This has the effect that the default list line will always be line one. However, to list subsequent lines, simply type List 2. Remember to delete the loader when entering has been completed.

Typing in the code using the loader of figure 4 should not take more than about 10 or 15 minutes, forbidding though the dissassembly looks. As bytes are entered, the format of figure 1 is reproduced on the screen to allow straightforward checking of each entry as it is made.

Figure 3 lists the massive Basic part of the program. Do not be tempted to add frills; they
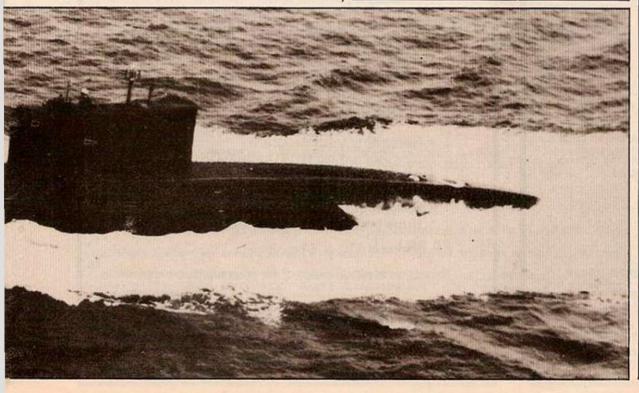
Figure 1.

```
16514  118  118   42   12   64
16519  229   17    7    0   25
16524   34   14   64    6   64
16529  175  215   16  253    6
16534   32   62   10  215   16
16539  253    1    0    1  175
16544  215   11  120  177   32
16549  249  225   30   51   25
16554   34   60   64  205  135
16559   65   62  255   50   70
16564   64   42  123   64   35
16569   34  123   64  126   33
16574   70   64   52   35  230
16579    7   60   60   50   62
16584   64  198    7  119   71
16589   14   26  205  245    8
16594   42   14   64   34   64
16599   64   22    0    0    0
16604   42   64   64  229    6
16609    5  113   35   16  252
16614  225  126   34   64   64
16619  126  254  118   40  197
16624    6    5   54  131   35
16629   16  251   43   43   43
16634   54  174   58   52   64
16639  230   30   79    6    8
16644  205  245    8   42   14
16649   64   22    0  126  254
16654   10   62   10   32    1
16659   61  119    6    2  197
16664   64   60   64   58   37
16669   43  254  247   32   12
16674   43  126  118   40  205
16679   19   34   60   64   32
16684  135   65  254  239   52
16689   17   35  229   30    7
16694   25  126  254  118  225
16699   40    9   34   60   64
16704  205  135   65  193   16
16709  209   24    2   24  146
16714  237   75   66   64   42
16719   68   64  203   65   32
16724   18   42   60   64   30
16729   36   25   58   37   64
16734  254  253   32   21   12
16739   58   62   71   16    3
16744   30   33   25   16    3
16749   12   24    7  126  254
16754  174   54   52   40   60
16759  237   67   66   64   14
16764   68   64    6   60   13
16769   32  253   16  251   24
16774  193    1  118    7   30
16779   33   42   60   64  229
16784  229   43  126  195   40
16789    5  114  167  237   82
16794  114  225   62  128  119
16799   35   16  252  126  185
16804   40    1  114  225  167
16809  237   82   54  135   35
16814   54  130   35  114  201
16819    1   23   52   42   64
16824   64   43  113   35   54
16829   39   35  112   35  112
16834   35   54   50   35  113
16839    1    0    0  205  245
16844    8  237   75   70   64
16849    6    0  201
```

will almost certainly lead to an out of memory report.

Keyboard decoding has been arranged to suit the destroyer skipper wearing regulation Arctic-issue mittens: any key 1 through 5 will move the ship left, while any key 6 to 0 will move it to the right. Pressing F releases a depth charge. As normal, these devices are set to trigger at the supposed depth of the submarine. This particular destroyer has excellent sonar; its charges always go off at the appropriate depth. They must, however, drop into the conning tower to produce the desired effect. To alter the speed at which events take place, try Poking different values into address 16767.

Figure 2.

```
U-BOAT HUNT DISASSEMBLY

INITSCRN   LD HL,
           (DISPLAY FILE)
           PUSH HL
           LD DE,7
           ADD HL,DE
           LD (DFCC),HL
           LD B,64
ZEROACC    XOR A
           RST 10H
           DJNZ ZEROACC
           LD B,4
           LD A,CODE "▆"
PRTACC     RST 10H
           DJNZ PRTACC
           LD BC,256
BLANKLNS   XOR A
           RST 10H
           DEC BC
           LD A,B
           OR C
           JR NZ,BLANKLNS
INITSHIP   POP HL
           LD E,51
           ADD HL,DE
           LD (SHIPAD),HL
           CALL PRTSHIP
           LD A,255
           LD (SUBCOUNT),A
NEWSUB     LD HL,(RAND)
           INC HL
           LD (RAND),HL
           LD A,(HL)
           LD HL,SUBCOUNT
           INC (HL)
           INC HL
           AND 7
           INC A
           INC A
           LD (SUBLINE),A
           ADD 7
           LD (HL),A
           LD B,A
           LD C,26
           CALL PRTPOS
           LD HL,(DFCC)
           LD (SUBAD),HL
           LD HL,(SUBAD)
PRTSUB     PUSH HL
           LD B,5
           LD (HL),C
           INC HL
           DJNZ -4
           POP HL
           DEC HL
           LD (SUBAD),HL
           LD A,(HL)
           CP NEWLINE
           JRZ NEWSUB
           LD B,5
           LD (HL),CODE "▄"
           INC HL
           DJNZ -5
           DEC HL
           DEC HL
           DEC HL
           LD (HL),CODE "▇"
WAVES      LD A,(FRAMES)
           AND 30
           LD C,A
           LD B,8
           CALL PRTPOS
           LD HL,(DFCC)
           LD D,0
           LD A,(HL)
           CP "▄"
           LD A,CODE "▄"
           JR Z,1
           DEC A
           LD (HL),A
MOVESHIP
ACTION?    LD B,2
           PUSH BC
           LD HL,(SHIPAD)
           LD A,(LAST-K)

LEFT ?     CP 247
           JR NZ,RIGHT ?
           DEC HL
           LD A,(HL)
           CP NEWLINE
           JR Z,DCHARGE
           LD (SHIPAD),HL
           CALL PRTSHIP
RIGHT ?    CP 239
           JR NZ,DCHARGE
           INC HL
           PUSH HL
           LD E,7
           ADD HL,DE
           LD A,(HL)
           CP NEWLINE
           POP HL
           JR Z,DCHARGE
           LD (SHIPAD),HL
           CALL PRTSHIP
           POP BC
           DJNZ ACTION?
           JR DCHARGE
JUMP       JR PRTSUB
DCHARGE    LD BC,(DCIND)
           LD HL,(DCAD)
           BIT 0,C?
           JR NZ,BLANKDC
           LD HL,(SHIPAD)
           LD E,36
           ADD HL,DE
           LD A,(LAST-K)
DCHARGE?   CP 253
           JR NZ,STODC
           INC C
           LD A,(SUBLINE)
           LD B,A
BLANKDC    LD (HL),D
           LD E,33
           ADD HL,DE
           DJNZ PRTDC
           INC C
           JR STODC
PRTDC      LD A,(HL)
           CP CONNINGTOWER
           LD (HL),CODE "O"
           JR Z,EXIT
STODC      LD (DCIND),BC
           LD (DCAD),HL
DELAY      LD B,MAINCOUNT
LOOP       DEC C
           JR NZ,LOOP
           DJNZ LOOP
           JR JUMP
PRTSHIP    LD BC,NEWLINE
           /SHIPLEN
           LD E,LINEINC
           LD HL,(SHIPAD)
           PUSH HL
           PUSH HL
           DEC HL
           LD A,(HL)
           CP C
           JR Z,5
           LD (HL),D
           AND A
           SBC HL,DE
           LD (HL),D
           POP HL
           LD A,CODE "▇"
PRTBLK     LD (HL),A
           INC HL
           DJNZ PRTBLK
           LD A,(HL)
           CP C
           JR Z,1
           LD (HL),D
           POP HL
           SBC HL,DE
           LD (HL),"F"
           INC HL
           LD (HL),"L"
           INC HL
           LD (HL),D
           RET
EXIT       LD BC,*/0
           LD HL,(SUBAD)
           DEC HL
           LD (HL),"*"
           INC HL
           LD (HL),"B"
           INC HL
           LD (HL),"O"
           INC HL
           LD (HL),"O"
           INC HL
           LD (HL),"M"
           INC HL
           LD (HL),"*"
           LD BC,0
           CALL PRTPOS
           LD BC,(SUBCOUNT)
           LD B,0
           RET
```

Figure 3.

```
BASIC PROGRAMME
10 PRINT USR 16516;
   " SUBS ESCAPED"
15 INPUT A$
20 RUN
```

Figure 4.

```
LOADER ROUTINE
10 FOR N=1 TO 338 STEP 5
15 SCROLL
20 PRINT 16513+N;
25 FOR K=0 TO 4
30 INPUT I
35 POKE 16513+N+K,I
40 PRINT TAB (7+K*5);I;
45 IF N+K=338 THEN STOP
50 NEXT K
55 PRINT
60 NEXT N
```

BREAK THROUGH THE BBC OPERATING SYSTEM

BBC

Chris Melville's last two articles delved into the BBC operating system — now you can break out.

THE PROGRAM published here is a Basic-driven assembly language program to play either Breakout or Breakthrough. It uses pixel graphics in Mode 7 so that it should run on a model A if you miss out all the comments and put multiple assembler statements per line. Using pixels in mode 7 needs some fiddling which would slow down a Basic program too much to make a challenging game, but this one is both fast and smooth. It was intended as an example of how the MOS calls would be used in games programming. There is not a direct memory reference in the whole program — unusual for a game — so the program will run on any operating system and also with the 6502A second processor installed. The features are: selectable bat size, increasing game speed as score increases, selectable initial hardness, selectable bat speed relative to ball, selectable number of balls per game, wall lowers every time you get through the last one, breakout or breakthrough, and high score.

Some of the operating system calls are as follows:

Purpose	Line numbers	MOS routine
Produce a beep	680-710	OSWORD(7)
Perform Tab	950-1000	OSWRCH(31)
Examine character at cursor position.	1020-1030	OSBYTE(135)

The Breakout program.

```
1ONERROR RUN
10*TV255,0
20MODE7
30 PROCSETUP
40PROCCOURT
50CALLPIP
60PRINTTAB(24,0); !S%; :IF?BC<54 AND?BY<21 GOTO50
70TIME=0:REPEAT SOUND &12,-8,TIME,5: UNTIL TIME>200
80SOUND&11,-15,0,5:IF ?BC=54 ?XV=?XV *(2*RND(2)-3):?YV=255:GOTO40
90?YV=0-ABS(?YV):BL%=BL%-1:PRINTTAB(37,0)BL%
100IF BL%>0 ?XV=ABS(?XV):GOTO50
110 IF !S%>HI_SCORE HI_SCORE=!S%
120PRINTTAB(3,16)CHR$133"HIT 'G'-ANOTHER GAME  OR"CHR$151;TAB(2,17)CHR$134"'ES
CAPE'-START FROM BEGINNING"CHR$151;
130IF NOT INKEY(-84) GOTO130
140?XV=(2*RND(2)-3):?YV=255:?WS=1:BL%=BALLS%:!(S%)=0:GOTO40
150DEF PROCCOURT:LOCAL I%,J%
155CLS
160IF ?WS<11 AND ?BC=54 ?WS=?WS+2
170CLS:PRINT:FOR I%=0 TO 22:PRINTCHR$152:NEXT
180PRINTTAB(1,1)STRING$(38,CHR$112)
190
200FORI%=2 TO23:PRINTTAB(1,I%)CHR$181;TAB(38,I%)CHR$234;:NEXT
210FORI%=?WS+2 TO ?WS+7:PRINTTAB(2,I%)STRING$(36,CHR$124):NEXT
220FORI%=1 TO ?WS +1:PRINTTAB(0,I%)CHR$151:NEXT
230RESTORE:FOR I%=1 TO 6:READJ%:PRINTCHR$J%:NEXT
240REPEAT:PRINTCHR$151:UNTIL  VPOS=23
250 DATA147,150,146,149,145,148
260T%=RND(15)+2
270?BC=0:?BP=19:?PB=39:?DB=1:PRINTTAB(19,21); CHR$42;STRING$(?BL-1,CHR$47);CHR$
37
280?BX=T%:?LX=T%:?BY=20:?LY=20:?PX=T%*2:?PY=60:?DX=0:?DY=0:PRINTTAB(T%,20)CHR$
33
290PRINTTAB(1,0)CHR$131;"HI_Score:";CHR$135;HI_SCORE;TAB(16,0)CHR$133;"Score:"
;CHR$134;!S%;TAB(29,0)CHR$130;"Balls:";CHR$131;BL%
300REPEAT UNTIL INKEY(-99)
310ENDPROC
320DEF PROCSETUP
330OSWORD=&FFF1:OSWRCH=&FFEE:OSBYTE=&FFF4
340VDU23;8202;0;0;0;
350DIM DOT 5,T1 0,T2 0,T3 0,T4 0,BX 0,BY 0,PX 0,PY 0,XV 0,YV 0,BP 0,PB 0,BL 0,
```

```
BV 0,BC 0,BEEP 8,WS 0,S% 3,DX 0,DY 0,DB 0,LX 0,LY 0,SPEED 0,THRU 0,LEVEL 15
   360?DOT=161:?(DOT+1)=162:?(DOT+2)=164
   370?(DOT+3)=168:?(DOT+4)=176:?(DOT+5)=224
   380!(BEEP)=&FFF10011:!(BEEP+4)=&5000F
   390?SPEED=30:!S%=0:?WS=1:?XV=1:?YV=255
   400PITCH=BEEP+4:HI_SCORE=0
   410 PROCASSEMBLE
   420PROCINSTRUCTIONS
   430ENDPROC
   440DEF FNGETBETWEEN(I%,J%):LOCAL K%
   450 PRINT" (";I%;" to ";J%;")";CHR$135;
   460REPEAT K%=GET-ASC"0":UNTIL K%>=I% AND K%<=J%
   470PRINTCHR$(K%+ASC"0")
   480=K%
   490DEF PROCINSTRUCTIONS:CLS
   500CLS:PRINT'CHR$131,"B R E A K O U T         C.MELVILLE '83"
   510PRINTCHR$145,STRING$(15,"\")SPC(9)STRING$(15,"\")
   520PRINT'CHR$134;"Input no. of balls per game ";:BALLS%=FNGETBETWEEN(1,5):BL%=
BALLS%
   530PRINT'CHR$130;"Input initial bat size";:?BL=FNGETBETWEEN(1,5)
   540PRINT'CHR$133;"Input bat speed ";:?BV=FNGETBETWEEN(1,4)
   550PRINT'CHR$130;"Hit O for breakOUT,T for breakTHROUGH":REPEAT X$=GET$:UNTIL
X$="O" OR X$="T":IF X$="O" ?THRU=0 ELSE ?THRU=255
   560PRINT'CHR$134;"Input initial game speed";:H%=FNGETBETWEEN(1,3)
   570 PRINT''CHR$129"Z-bat left,X-bat right, SPACE BAR-serve";
   580 PRINT'CHR$136;"      key  to  start ...."
   590 X=GET
   600IF H%=1 !(LEVEL)=&1E23282B:!(LEVEL+4)=&18191C1C:!(LEVEL+8)=&12131416:!(LEVE
L+12)=&080C0F11 ELSE!(LEVEL)=&18191C1C:!(LEVEL+4)=&12131416:!(LEVEL+8)=&080C0F11
:!(LEVEL+12)=&080C0F11
   610IFH%=3 !(LEVEL)=&12131416:!(LEVEL+4)=&080C0F11:!(LEVEL+8)=&080C0F11:!(LEVEL
+12)=&080C0F11
   620ENDPROC
   630DEF PROCASSEMBLE:LOCAL PASS%
   640DIM FREE%2000
   650FOR PASS%=0 TO 2 STEP2
   660P%=FREE%
   670E OPT PASS%
   680.PIP LDX #BEEP MOD 256
   690     LDY #BEEP DIV 256
   700     LDA #7        \SET UP OSWORD
   710     JSR OSWORD    \DO A BEEP
   720.BALL LDA PX       \FIND NEW
   730JSR DELAY
   740     CLC           \PIXEL POS.
   750     ADC XV        \OF BALL BY
   760     STA PX        \ADDING ON
   770     LDA PY        \X,Y VELOCITY
   780     CLC           \COMPONENTS
   790     ADC YV        \TO OLD POS.
   800     STA PY
   810     LDA PX        \WORK OUT TAB
   820     LSR A         \POSITION OF
   830     STA BX        \BALL
   840
   850
   860     LDA PY
   870     LDY #255      \WORK OUT TAB
   880.SUB INY:SEC       \POS. OF Y
   890     SBC #3        \COORD. OF
   900     BCS SUB       \BALL AND ITS
   910     CLC:ADC #3    \PIXEL HEIGHT
920 STY BY             \ABOVE Y-CD.
   930     STA DY
   940
   950     LDA #31       \CODE FOR TAB
   960     JSR OSWRCH
   970     LDA BX        \MOVE TO X,Y
   980     JSR OSWRCH    \POS.OF BALL
   990     LDA BY
  1000     JSR OSWRCH
  1010
  1020     LDA #135      \EXAMINE CHAR
  1030     JSR OSBYTE    \ALREADY HERE
  1040     STX T1        \ + PUT IN T1
  1050
  1060     LDA T1        \SEE IF IT IS
  1070     CMP #112      \PART OF TOP
  1080     BNE SKP1      \WALL,IF SO
  1090     JMP LID       \GO TO "LIC"
  1100
  1110.SKP1 CMP #181     \SEE IF IT IS
  1120     BNE SKP2      \PART OF LEFT
  1130     JMP SIDE      \SIDE WALL...
  1140
  1150.SKP2 CMP #234     \AND SAME FOR
  1160     BNE SKP3      \RIGHT WALL..
  1170     JMP SIDE
  1180
  1190.SKP3 CMP #47      \IF IT WAS
  1200     BNE SKK2      \PART OF BAT
  1210     JMP BAT       \GO TO "BAT"
  1220
  1230.SKK2 CMP #42      \ALSO TEST FOR
  BNE SKK1              \A HIT ON THE
  1250     JMP LBAT      \SIDE OF BAT
  1260
 -1270.SKK1 CMP #37      \AND OTHER ONE
  1280     BNE SKK3
  1290     JMP RBAT
  1300
  1310.SKK3 LDA BY       \IF BALL HAS
  1320     CMP #21       \GONE OUT,
  1330     BNE SKP4      \RETURN TO
  1340     RTS           \BASIC.
  1350
  1360.SKP4 LDA T1       \SEE IF BALL
  1370     CMP #124      \HAS HIT A
```

```
  1380     BNE SKP5      \BRICK,IF SO
  1390     JMP BRICK     \GOTO"BRICK"
  1400
  1410.SKP5 LDA #31      \MOVE TO TAB
  1420     JSR OSWRCH    \POS. OF LAST
  1430     LDA LX        \POS. OF BALL
  1440     JSR OSWRCH    \AND RUB IT
  1450     LDA LY        \OUT
  1460     JSR OSWRCH
  1470     LDA #ASC" "
  1480     JSR OSWRCH
  1490
  1500     LDA PX        \FIND PIXEL
  1510     AND #1        \INSIDE X-CD.
  1520     STA DX        \OF BALL.
  1530
  1540     CLC:ADC DY    \USE LOOK-UP
  1550     CLC:ADC DY    \TABLE TO
  1560     TAX           \DECIDE WHICH
  1570     LDA.DOT,X     \GRAPHICS CHR
  1580     STA T2        \IS REQUIRED.
  1590
  1600     LDA #31       \DO A TAB,AND
  1610     JSR OSWRCH    \PRINT THE
  1620     LDA BX        \GRAPHICS
  1630     JSR OSWRCH    \CHARACTER
  1640     LDA BY        \THAT WAS
  1650     JSR OSWRCH    \WORKED OUT
  1660     LDA T2        \TO REPRISENT
  1670     JSR OSWRCH    \THE BALL.
  1680
  1690     LDA BX:STA LX \SET NEW LAST
  1700     LDA BY:STA LY \COORDINATES.
  1710
  1720\THE NEXT SECTION IS RESPONSIBLE
  1730\FOR SCANNING THE KEYBOARD AND
  1740\MOVING BAT ACCORDINGLY.
  1750
  1760.MBAT LDA #129     \TEST Z,X KEYS
  1770     LDY #&FF      \USING THE OS
  1780     LDX #&8D      \OSBYTE 129
  1790     JSR OSBYTE    \EQUIVALENT OF
  1800     TYA:BMI XGOT  \INKEY(-67)
  1810     LDA #129      \OR INKEY(-98)
  1820     LDY #&FF      \AND BRANCH TO
  1830     LDX #&9E      \CORRESPONDING
  1840     JSR OSBYTE    \BAT-MOVING
  1850     TYA           \ROUTINES IF
  1860     BMI ZGOT      \Z,X PRESSED
  1870
  1880.KBD2 JMP BALL     \RETURN TO START.
  1890
  1900.XGOT LDA BV:STA T1 \FIND VALUE
  1910     JMP BT1       \TO BE ADDED
  1920.ZGOT LDA #0:SEC    \TO BAT POS.
  1930     SBC BV        \AND STORE
  1940     STA T1        \IT IN T1.
  1950
  1960.BT1 LDA #31       \MOVE TO TAB
  1970     JSR OSWRCH    \POS.OF START
  1980     LDA BP        \OF BAT.
  1990     JSR OSWRCH
  2000     LDA #21
  2010     JSR OSWRCH
  2020
  2030     LDY BL        \NEXT FEW
  2040     LDA #ASC" "   \LINES DEALS
  2050.BT2 JSR OSWRCH    \WITH RUBBING
  2060     DEY:BNE BT2   \OUT OLD BAT.
  2070     LDA DB
```

```
  2080     BEQ BT3
  2090     LDA #ASC" ":JSR OSWRCH
  2100
  2110.BT3 LDA PB        \NEXT SECTION
  2120     CLC:ADC T1    \WORKS OUT
  2130     STA T2        \POSITIONAL
  2140     AND #1:STA T3 \DETAILS OF
  2150     LDA T2:LSR A  \NEW BAT AND
  2160     STA T4        \STORES THEM.
  2170
  2180     CLC           \WORK OUT END
  2190     ADC BL        \POS.OF BAT
  2200     CLC           \AND SEE IF
  2210     ADC T3        \TOO FAR TO
  2220     CMP #39       \RIGHT#(IF SO
  2230     BCC BTT       \DO NEXT BIT)
  2240     LDA #38       \MOVE BAT TO
  2250     SEC:SBC BL    \RIGHTMOST
  2260     STA BP        \POSITION
  2270     ASL A:STA PB
  2280     LDA #0:STA DB
  2290     JMP BT4
  2300
  2310.BTT LDA T4        \HAS BAT HIT
  2320     CMP #2        \WALL AT LEFT
  2330     BCS BTO       \NO-SKIP
  2340     LDA #4:STA PB
  2350     LDA #0:STA DB
  2360     LDA #2:STA BP
  2370     JMP BT4
  2380
  2390.BTO LDA T2:STA PB \TRANSFER
  2400     LDA T3:STA DB \TO PERMANENT
  2410     LDA T4:STA BP \REGISTERS.
  2420
  2430.BT4 LDA #31       \DRAW BAT AT
  2440     JSR OSWRCH    \POSITION
2450 LDA BP             \GIVEN BY THE
  2460     JSR OSWRCH    \CONTENTS OF
  2470     LDA #21       \THE PERMNT.
  2480     JSR OSWRCH    \REGISTERS.
  2490     LDA DB
  2500     BEQ BT5
  2510     LDA #42:JMP BT6
  2520.BT5 LDA #47
  2530.BT6 JSR OSWRCH
  2540     LDY BL:LDA #47
  2550.BT7 DEY:BEQ BT8
  2560     JSR OSWRCH
  2570     JMP BT7
  2580.BT8 LDA DB
  2590     BEQ BT9
  2600     LDA #37:JSR OSWRCH
  2610.BT9 JMP BALL \RETURN TO START
  2620
  2630
  2640\THE NEXT BIT IS THE ROUTINE FOR
  2650\WHEN THA BALL HITS THE "ROOF"
  2660
  2670.LID LDA PX:CMP #4:BNE LD0
  2680     LDA PY:CMP #6:BNE LD0
  2690     LDA #0:SEC:SBC XV:STA XV
  2700.LD0 LDA #150      \SET UP PITCH
  2710.LD1 STA PITCH     \TO BE BEEPED
  2720     LDA #0        \NEGATE AND
  2730     SEC:SBC YV    \RE-STORE Y
  2740     STA YV        \COMPONENT OF
  2750     JMP PIP       \BALL VELOITY
  2760
  2770\AND WHEN THE BALL HITS THE SIDE:
  2780
  2790.SIDE LDA #90      \AS BEFORE...
  2800     STA PITCH
  2810     LDA #0
  2820     SEC:SBC XV
  2830     STA XV
  2840     LDA BY
  2850     CMP #21
  2860     BCC SID1
  2870     LDA PB
  2880     CMP #5
  2890     BCC SID2
  2900     JMP SKK3
  2910.SID2 LDA #0:SEC:SBC YV:STA YV
  2920.SID1 JMP PIP
  2930
  2940\AND WHEN THE BALL HITS THE BAT:
  2950
  2960.BAT LDA XV        \BALL LEAVES
  2970     BMI B1        \MIDDLE OF BAT
```

Perform Tab
Test z and x keys
on keyboard   1760-1790   OSBYTE(129)
Printing spaces to
rub out bat   2030-2090   OSWRCH

They are repeated at several different places in the program. The assembly code is fully commented so it should be possible to develop the game further if required.

```
2980        LDY #1        \WITH UNIT X
2990        STY XV        \VELOCITY IN
3000        JMP B2        \MAGNITUDE
3010.B1     LDY #255
3020        STY XV
3030.B2     LDA #10
3040        JMP LD1
3050
3060.RBAT   LDA #2        \IF BALL HIT
3070        STA XV        \EDGE OF BAT
3080        JMP B2        \DIFFERENT
3090.LBAT   LDA #254      \BOUNCING
3100        STA XV        \OCCURS
3110        JMP B2
3120
3130\THE NEXT SECTION DEALS WITH THE
3140\BALL HAVING HIT A BRICK.
3150
3160.BRICK  LDA #8        \WORK OUT THE
3170        SEC           \ROW NUMBER
3180        SBC BY        \AND PUT IT
3190        CLC           \IN T1.
3200        ADC WS
3210        STA T1
3220
3230        LDA BX        \NOW WORK OUT
3240        SEC:SBC #2    \THE START
3250        AND #&FC      \POS. OF BRICK
3260        CLC:ADC #2
3270        STA T2        \ =T2
3280
3290        LDA #31       \DO A TAB TO
3300        JSR OSWRCH    \THEBEGINNING
3310        LDA T2        \OF THE BRICK
3320        JSR OSWRCH
3330        LDA BY
3340        JSR OSWRCH
3350
3360        LDA #ASC" "   \NOW BLANK OUT
3370        JSR OSWRCH    \BRICK
3380        JSR OSWRCH
3390        JSR OSWRCH
3400        JSR OSWRCH
3410
3420        LDA T1        \WORK OUT
3430        TAX           \PITCH OFBEEP
```

```
3440        ASL A
3450        ASL A
3460        ASL A
3470        ASL A
3480        STA PITCH
3490
3500        TXA           \INCREMENT
3510        CLC:ADC S%    \SCORE AS PER
3520        STA S%        \ROW HIT.
3530        LDA #0        \NOTE-SCORE
3540        ADC S%+1      \IS A 2-BYTE
3550        STA S%+1      \QUANTITY
3560        JSR PRMS      \THIS ROUTINE CHECKS HOW HIGH THE SCORE IS AND
CHANGES THE DIFFICULTY ACCORDINGLY.
3570
3580        LDA THRU      \NEGATE Y-CMP
3590        BMI IBC       \OF VELOCITY
3600        LDA #0        \IF NOT
3610        SEC:SBC YV    \BREAKTHROUGH
3620        STA YV
3630
3640.IBC    LDA BC
3650        CLC:ADC #1    \INCREMENT
3660        STA BC        \BRICK COUNT
3670
3680        LDA #0:TAY:TAX:RTS  \RETURN TO BASIC WITH ALL REGISTERS ZERO'D
3690
3700\ THE FOLLOWING SUBROUTINE USES A LOOK UP TABLE CALLED "LEVEL" TO DECIDE
ON THE SPEED OF THE GAME FOR ANY PARTICULAR SCORE LESS THAN 1024
3710.PRMS   LDX S%+1      \GET SCOREHIGH
3720        LDY S%        \GET SCORE LOW
3730        TXA:ROR A
3740        TAX:TYA
3750        ROR A:TAY:TXA
3760        ROR A:TAX:TYA
3770        ROR A:TAY
3780        LSR A:LSR A:LSR A:LSR A
3790        TAX:LDA LEVEL,X:STA SPEED:RTS
3800.DELAY  PHA:TYA:PHA:TXA:PHA
3810        LDX SPEED
3820.D2     LDY #255
3830.D1     DEY:BNE D1
3840        DEX:BNE D2
3850        PLA:TAX:PLA:TAY:PLA:RTS
3860]
3870NEXT PASS%
3880ENDPROC
```

■

```
FILL
BACKGROUND
(ASCII CODE- )
RST 24          223
LD HL (9216)    33     0    36
LD B (23)        6    23
PUSH BC         197
LD B (32)        6    32
LD A (HL)       126
CP (32)         254   32
JRNZ (+2)        32    2
LD A E          123
LD (HL) A       119
INC HL           35
DJNZ (-10)       16   246
POP BC          193
DJNZ (-16)       16   240
JP (IY)         253   233

FILL
FOREGROUND
(ASCII CODE- )
RST 24          223
LD HL (9216)    33     0   36
LD B (23)        6    23
PUSH BC         197
LD B (32)        6    32
LD A (HL)       126
CP (32)         254   32
JRZ (+2)         40    2
LD A E          123
LD (HL) A       119
INC HL           35
DJNZ (-10)       16   246
POP BC          193
DJNZ (-16)       16   240
JP (IY)         253   233

SEARCH&
REPLACE
(ASCII CODE SEARCH
 ASCII CODE REPLACE- )
RST 24          223
LD C E           75
RST 24          223
LD HL (9216)    33     0   36
LD B (23)        6    23
PUSH BC         197
LD B (32)        6    32
LD A (HL)       126
CP E            187
JRNZ (+2)        32    2
LD A C          121
LD (HL) A       119
INC HL           35
DJNZ (-9)        16   247
POP BC          193
DJNZ (-15)       16   241
JP (IY)         253   233

INVERT
( - )
LD HL (9216)    33     0   36
LD B (23)        6    23
PUSH BC         197
LD B (32)        6.   32
LD A (HL)       126
ADD A (128)     198   128
LD (HL) A       119
INC HL           35
DJNZ (-7)        16   249
POP BC          193
DJNZ (-13)       16   243
JP (IY)         253   233
```

ACES CAN GO FASTER

Forth is fast, but a little bit of machine code will take you through the light barrier. Simon Cross shows you how to accelerate your Jupiter Ace with this handy kit of 10 screen-handling routines for those situations where Forth is still not quite fast enough and the ultimate speed of machine code is what you require.

THE JUPITER ACE offers an alternative for those who wish to write fast arcade games but cannot face the sometimes daunting challenge presented by machine code. The speed advantage that Forth has over Basic enables the games programmer to write "space-invader" type programs without reaching for an assembler. However, there are still some situations where Forth is not quite fast enough and the ultimate speed of machine code is required.

One such situation is writing screen-handling routines such as scrolling. Try this Forth word which fills the screen with a character specified by an ASCII code taken from the top of the stack:

```
: FILL
9952 9216
DO
```
```
DUP I C!
LOOP
DROP
;
```

Execute this word by typing an ASCII code followed by "fill", for example "42 fill" will produce a screenful of stars. It can be seen that although the screen is filled quite rapidly, the process of printing is visible passing in a wave from the top to the bottom of the screen. When this routine is written in machine code the process appears to be instantaneous; this is useful for creating explosion effects and the like. The difference in speed between Forth and machine code becomes more marked with more complex routines such as scrolling.

Even when routines have to be written in machine code the Forth environment still has

ORIC
SOFTWARE
from TANSOFT

ORIC CHESS

A superb chess game written in Basic with a Machine Code Move Search Routine. Features high resolution colour graphics, record of moves, allows Castling and En-Passant, 5 levels of play and computer can play black or white.

ONLY £9.99 inc VAT

48K only

ZODIAC

A 36K adventure game that will test your intelligence and patience for hours. Find the 12 hidden signs of the Zodiac to discover incredible treasures. Can you ward off the angry Yeti? What is the meaning of the radio that plays the 'Archers' over and over again (through your loudspeaker!)?

Only £9.99 inc VAT

48K only

ORIC FLIGHT

Can you fly your new 787 jet on to the runway without crashing, stalling or running out of fuel? Full digital instrument readout plus cockpit view of final runway approach. Superb graphics and sound.

Only £9.99 inc VAT

48K only

Please send me	Quantity	Total
Oric-Chess at £9.99		
Zodiac at £9.99		
Oric-Flight at £9.99		
TOTAL		

Postage per cassette is 40p

Name _____

Address _____

_____

_____

Please send me latest price list _____

Please allow 28 days for delivery

All cheques payable to
TANSOFT LTD
3 CLUB MEWS
ELY
CAMBS
CB7 4NW

Tansoft is the software division of
Tangerine Computers.

some advantages. The routine can initially be written in Forth to test its function in the context of the complete program, particularly its effects on the stack. If the routine works it can be translated into machine code without any alterations to the rest of the program. The routine is called by its name which is more convenient than Rand Usr followed by the address, the technique some Basics use. The main advantage is that since Forth words can be moved around the dictionary by Redefine and Load the machine code must be relocatable.

This article describes 10 screen-handling words written in machine code for the Jupiter Ace. Any combination of them may be included in ordinary Forth programs where speed is important. No knowledge of machine code is required to use them since once they have been entered they act like any other words in the Forth dictionary. The simplest way to enter the code is by using the defining word "Code" described on page 147 of the Jupiter Ace manual:

```
        DEFINER CODE
        DOES>
            CALL
        ;
```

The machine code is entered by typing Code then the name of the word followed by the decimal code, each byte being separated by C,. For example the Scrollup routine would be entered by typing:

```
CODE SCROLLUP 33 C, 32 C, 36 C,
            17 C, 0 C, etc.
```

On pressing Enter the whole typed section will be copied to the upper screen and OK will appear after it. The word can then be executed by typing its name, for example:

```
        SCROLLUP
```

Code is not a very user-friendly word but, since it is not possible to delete it from the dictionary when the code has been entered, it is important that it should not take up much memory space.

The 10 words will fit into the unexpanded 3K Jupiter Ace, but this does not leave much memory space in which to write programs to utilise the routines. It is better to only enter the words necessary for a specific program.

Here is a description of the individual words: Scrollup scrolls the entire screen one line up and blanks the bottom line; Scrolldown scrolls the entire screen one line down and blanks the top line; Scrollright scrolls the entire screen one column to the right and blanks the left-hand column; and Scrollleft scrolls the entire screen one column to the left and blanks the right-hand column.

Fillscreen fills the screen with a character specified by an ASCII code which should be put on the stack before the word is executed. Fillscreen uses the RST 24 routine in the ROM which takes off the top of the Forth stack and puts it in the DE register pair. Fillforeground fills all the non-blank areas of the screen with a character specified by an ASCII code on the stack. Fillbackground fills all the blank areas of the screen with a character specified by an ASCII code on the stack.

Search&replace takes two numbers from the stack. The first number is the ASCII code of the character to be searched for on the screen. Any occurrence of this character is replaced by another character whose ASCII code is given by the second number. This word is useful to selectively "flash" parts of the screen for explosions or countdowns.
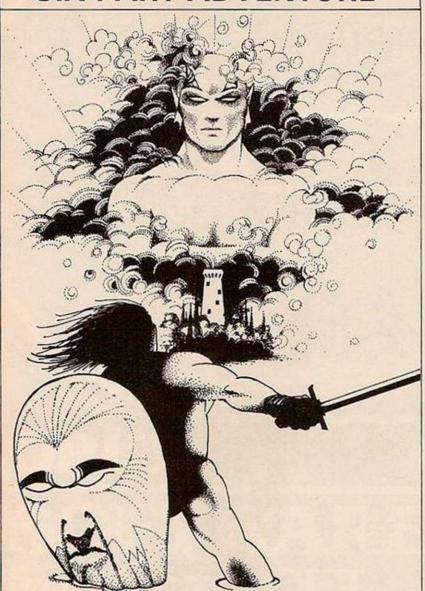
Invert turns all the characters on the screen into their inverse forms. It is useful for explosion effects when used in loops such as this:

```
    : BANG
    21 1
    DO
        INVERT I 10 * 60 BEEP
    LOOP
    ;
```

Border prints a border round the edge of the screen using the character specified by an ASCII code on the stack. ■

```
SCROLLUP
( - )
LD HL (9248)    33   32   36
LD DE (9216)    17    0   36
LD B (22)        6   22
PUSH BC        197
LD B (32)        6   32
LD A (HL)      126
EX DE HL       235
LD (HL) A      119
INC HL          35
EX DE HL       235
INC HL          35
DJNZ (-8)       16  248
POP BC         193
DJNZ (-14)      16  242
DEC HL          43
LD A (32)       62   32
LD B A          71
LD (HL) A      119
DEC HL          43
DJNZ (-4)       16  252
JP (IY)        253  233
```

```
SCROLLDOWN
( - )
LD HL (9919)    33  191   38
LD DE (9951)    17  223   38
LD B (22)        6   22
PUSH BC        197
LD B (32)        6   32
LD A (HL)      126
EX DE HL       235
LD (HL) A      119
DEC HL          43
EX DE HL       235
DEC HL          43
DJNZ (-8)       16  248
POP BC         193
DJNZ (-14)      16  242
INC HL          35
LD A (32)       62   32
LD B A          71
LD (HL) A      119
INC HL          35
DJNZ (-4)       16  252
JP (IY)        253  233
```

```
SCROLLRIGHT
( - )
LD HL (9950)    33  222   38
LD DE (9951)    17  223   38
LD B (23)        6   23
PUSH BC        197
LD B (31)        6   31
LD A (HL)      126
EX DE HL       235
LD (HL) A      119
DEC HL          43
EX DE HL       235
DEC HL          43
DJNZ (-8)       16  248
EX DE HL       235
LD A (32)       62   32
LD (HL) A      119
DEC HL          43
EX DE HL       235
DEC HL          43
POP BC         193
DJNZ (-21)      16  235
JP (IY)        253  233
```

```
SCROLLLEFT
( - )
LD HL (9217)    33    1   36
LD DE (9216)    17    0   36
LD B (23)        6   23
PUSH BC        197
LD B (31)        6   31
LD A (HL)      126
EX DE HL       235
LD (HL) A      119
INC HL          35
EX DE HL       235
INC HL          35
DJNZ (-8)       16  248
EX DE HL       235
LD A (32)       62   32
LD (HL) A      119
INC HL          35
EX DE HL       235
INC HL          35
POP BC         193
DJNZ (-21)      16  235
JP (IY)        253  233
```

```
FILLSCREEN
(ASCII CODE- )
RST 24         223
LD A E         123
LD HL (9216)    33    0   36
LD B (23)        6   23
PUSH BC        197
LD B (32)        6   32
LD (HL) A      119
INC HL          35
DJNZ (-4)       16  252
POP BC         193
DJNZ (-10)      16  246
JP (IY)        253  233
```

```
BORDER
(ASCII CODE- )
RST 24         223
LD A E         123
LD HL (9216)    33    0   36
LD B (32)        6   32
LD (HL) A      119
INC HL          35
DJNZ (-4)       16  252
LD DE (31)      17   31    0
LD B (21)        6   21
LD (HL) A      119
ADD HL DE       25
LD (HL) A      119
INC HL          35
DJNZ (-6)       16  250
LD B (32)        6   32
LD (HL) A      119
INC HL          35
DJNZ (-4)       16  252
JP (IY)        253  233
```

STRUCTURES FOR BBC

Woodward and Berry's card game makes learning structures fun.
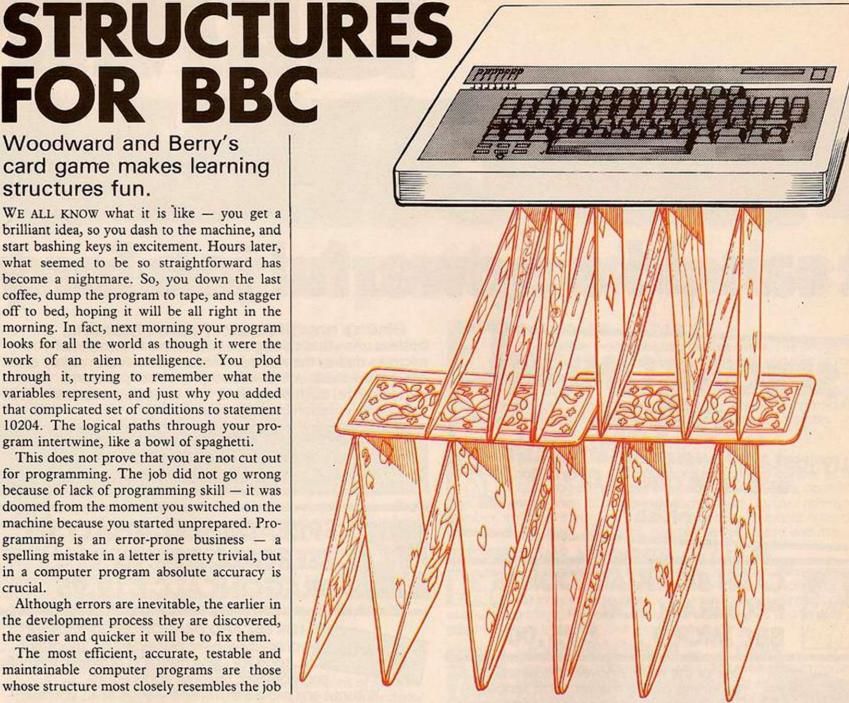
WE ALL KNOW what it is like — you get a brilliant idea, so you dash to the machine, and start bashing keys in excitement. Hours later, what seemed to be so straightforward has become a nightmare. So, you down the last coffee, dump the program to tape, and stagger off to bed, hoping it will be all right in the morning. In fact, next morning your program looks for all the world as though it were the work of an alien intelligence. You plod through it, trying to remember what the variables represent, and just why you added that complicated set of conditions to statement 10204. The logical paths through your program intertwine, like a bowl of spaghetti.

This does not prove that you are not cut out for programming. The job did not go wrong because of lack of programming skill — it was doomed from the moment you switched on the machine because you started unprepared. Programming is an error-prone business — a spelling mistake in a letter is pretty trivial, but in a computer program absolute accuracy is crucial.

Although errors are inevitable, the earlier in the development process they are discovered, the easier and quicker it will be to fix them.

The most efficient, accurate, testable and maintainable computer programs are those whose structure most closely resembles the job

HOUSE OF CARDS

that the program has been designed to do.

Writing programs that work, and completing them before you get fed up with them, is fun. Unless we pay due regard to the above maxim, that fun will be replaced by misery and frustration. So, what we want is a methodology for approaching this programming task — and structured programming is just that.

Let's go back to that wonderful inspirational moment when the idea comes. Instead of rushing to your machine and spoiling it all, grab a piece of paper and write down in plain language what the objective of the program is. The original statement is usually termed "the problem". Next, try to break down the main problem into subordinate problems, the essential things your program has to achieve to solve the problem. You can continue in this way, building a hierachy of problems at increasing levels of detail.

So far, this may all sound pretty trite — where is the methodology? There are many different schemes to be found in books and in

use, but they all attempt to show the same things: a hierachy of problems; decisions or branches; loops or iterations.

From now on, we will try to explain the process by developing a program to play the card game of Pelmanism. If you are not familiar with the game, do not worry — all will be revealed as we proceed. You will acquire the necessary knowledge as your analysis gets deeper into the problem. Approaching problem analysis in this layered way is called top-down design.

Figure 1 shows a possible first-level breakdown of the Pelmanism problem. The main problem, "Play Pelmanism", is found to consist of two sub-problems, "Set up game" and "Play Game". The diagram is read top-to-bottom, left-to-right, so the diagram tells you that you have to set up the game before you can play it.

We discover that set-up consists of getting a pack of playing cards, shuffling them, and then laying them all, face down, in random places on the table. Figure 2 shows how these

sequential tasks are added to the diagram.

That is probably as far as we need to go at present in the set-up problem area, so let us now analyse the play-game problem. We discover the rules are that two players take turns. At each turn, the player turns up two cards. If they are of the same value, for example two kings, the player gets a point, and the cards are removed.

If they are not of the same value, they are replaced in their original positions, face down. So, we have a loop — rotate around the players in turn — and a decision — if the cards match, do one thing, otherwise do something else. Figure 3 shows how the play-game problem can be broken down to incorporate these tasks, and includes the loop and the decision. The loops are represented by curved arrows. The decision is represented by a little diamond.

It should be apparent that what happens on Player B's turn is essentially the same as happens for Player A. We have revealed a whole chunk of problem which might have

GEMIN

PERFORMAN

It can do a powerful job for yo

Sinclair ZX Spectr

16K or 48K RAM... full-size moving-key keyboard... colour and sound... high-resolution graphics...

From only £125!

ZX Spectrum

BLUE EDIT · RED CAPS LOCK · MAGENTA TRUE VIDEO · GREEN INV. VIDEO · CYAN · YELLOW · WHITE

First, there was the world-beating Sinclair ZX80. The first personal computer for under £100.

Then, the ZX81. With up to 16K RAM available, and the ZX Printer. Giving more power and more flexibility. Together, they've sold over 500,000 so far, to make Sinclair world leaders in personal computing. And the ZX81 remains the ideal low-cost introduction to computing.

Now there's the ZX Spectrum! With up to 48K of RAM. A full-size moving-key keyboard. Vivid colour and sound. High-resolution graphics. And a low price that's unrivalled.

Professional power – personal computer price!

The ZX Spectrum incorporates all the proven features of the ZX81. But its new 16K BASIC ROM dramatically increases your computing power.
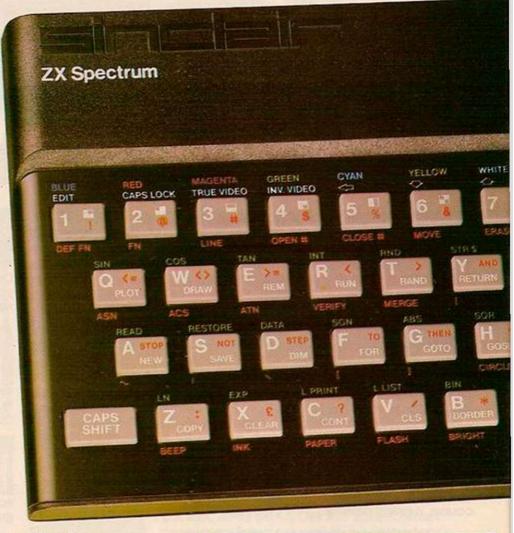
You have access to a range of 8 colours for foreground, background and border, together with a sound generator and high-resolution graphics.

You have the facility to support separate data files.

You have a choice of storage capacities (governed by the amount of RAM). 16K of RAM (which you can uprate later to 48K of RAM) or a massive 48K of RAM.

Yet the price of the Spectrum 16K is an amazing £125! Even the popular 48K version costs only £175!

You may decide to begin with the 16K version. If so, you can still return it later for an upgrade. The cost? Around £60.

Ready to use today, easy to expand tomorrow

Your ZX Spectrum comes with a mains adaptor and all the necessary leads to connect to most cassette recorders and TVs (colour or black and white).

Employing Sinclair BASIC (now used in over 500,000 computers worldwide) the ZX Spectrum comes complete with two manuals which together represent a detailed course in BASIC programming. Whether you're a beginner or a competent programmer, you'll find them both of immense help. Depending on your computer experience, you'll quickly be moving into the colourful world of ZX Spectrum professional-level computing.
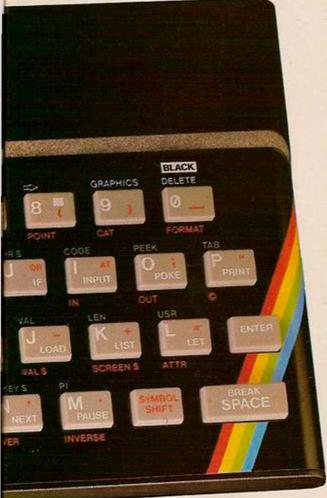
There's no need to stop there. The ZX Printer – available now – is fully compatible with the ZX Spectrum. And later this year there will be Microdrives for massive amounts of extra on-line storage, plus an RS232 / network interface board.

Key features of the Sinclair ZX Spectrum

- Full colour – 8 colours each for foreground, background and border, plus flashing and brightness-intensity control.

- Sound – BEEP command with variable pitch and duration.

- Massive RAM – 16K or 48K.

- Full-size moving-key keyboard – all keys at normal typewriter pitch, with repeat facility on each key.

- High-resolution – 256 dots horizontally x 192 vertically, each individually addressable for true high-resolution graphics.

- ASCII character set – with upper- and lower-case characters.

- Teletext-compatible – user software can generate 40 characters per line or other settings.

- High speed LOAD & SAVE – 16K in 100 seconds via cassette, with VERIFY & MERGE for programs and separate data files.

- Sinclair 16K extended BASIC – incorporating unique 'one-touch' keyword entry, syntax check, and report codes.

Let Commodore expand your horizons.

VIC 20 is the finest home computer that money can buy.

And the better you get to know it, the more confident, adventurous and ambitious you'll become.

You'll want to take advantage of the vast range of VIC software: a superb and constantly-growing selection of programs, embracing business systems, entertainment, education and many applications in the home.

Every program in the series has been designed by experts, and chosen for its quality and value for money.

VIC business software covers a wide range of applications, including spread-sheet analysis, stock control, information handling and word-processing.
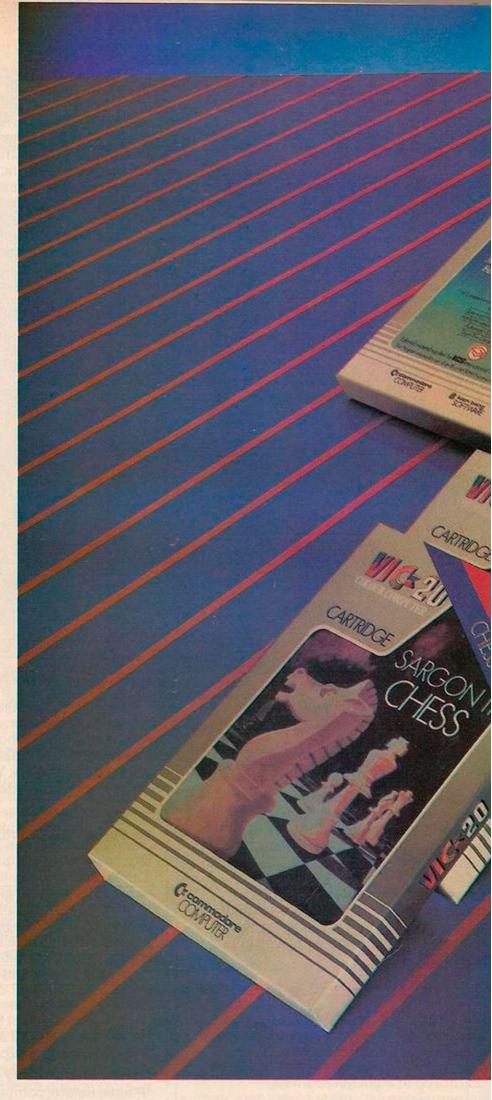
A mind-blowing range of games including Scott Adams' world-famous 'Adventure' series.

Advanced space games, including the sophisticated 'Omega Race'.

Learn subjects as diverse as English Language, programming, and biology.

And 'home' software ranges from IQ tests to Robert Carrier menus.

In addition, there is a range of VIC software, like programmers' aids and graphics packages – to add to your understanding and enjoyment of computers and computing.

There's even a special 'VicSoft' Club for VIC 20 enthusiasts, with many advantages including special offers to club members.

VIC software will expand your horizons. And your mind.

PRICES RANGE FROM £4.99 to £24.95 INC. VAT.

C= commodore VIC 20

Figure 1.

resulted in duplicate code, had we started to program too early. Of course, most of you would have spotted such duplication straight away, but the principle is valid. If it occurred at a lower level, you might not have recognised it. There is no need to duplicate parts of the diagram, so we simply put the note "as for player A" under "Player B's turn" — see Figure 3.

Just identifying a loop is hardly sufficient — we have to say what stops it. Once again, this is expressed in plain words — no flags to be used at this stage, please! Therefore, in Figure 3, we have added legends like "Repeat until all the cards have been paired" to the arrows representing the loops. Similarly, we have defined the decisions, but, because of lack of space, these definitions appear as footnotes. The other addition in Figure 3 is the box "Print Results". We have decided that, at the end of the game, we should display the scores.

We hope you can see already that this is an easy-to-understand representation of the game problem. You will notice that it describes the Problem, and not the Solution — so it is just as appropriate as a description of the game as played with cards as of its simulation on the computer. Therefore, when you do eventually code it, your program will represent reality, will correspond closely in structure to your design solution, and you will be able to enhance it later, if you wish.

Notice also that the structure diagram is universal, and is not tied to any particular programming language or dialect, or to any particular computer. For example, we have not said whether the loops will be implemented as "For-Next", "Repeat-Until" or "Do-While".

Now, although we have some way to go to fully define the problem, at the level of detail required for a complete solution, we could at this stage start to write the program. The

Figure 2.

essence of the structured "top-down" approach is that it does make it possible to develop programs in a modular way. Let us take the low-level boxes, such as "Pick a card". They are still black boxes at this stage, and can be coded as dummy procedures or sub-routines. Such dummies are known in the trade as "stubs". You can then run and test your skeleton program for basic errors. If you include temporary Print statements within your "stubs", you can prove to yourself that the procedures or sub-routines are being entered and left in the correct sequence. Note that you may have to force the program to proceed from one state to another by other temporary statements. Take the following example:

```
  10 PROCA (or GOSUB 1010)
  20 FLAG=0
  30 REPEAT
  40 PROCB (or GOSUB 2010)
  50 UNTIL FLAG=1
  60 PROCC (or GOSUB 3010)
  70 END
1000 DEF PROCA (not required for GOSUB)
1010 PRINT "PROCA HAS BEEN EXECUTED"
1999 ENDPROC (or RETURN)
2000 DEF PROCB (not required for GOSUB)
2010 PRINT "PROCB HAS BEEN EXECUTED"
2999 ENDPROC (or RETURN)
3000 DEF PROCC (not required for GOSUB)
3010 PRINT "PROCC HAS BEEN EXECUTED"
3999 END PROC (or RETURN)
```
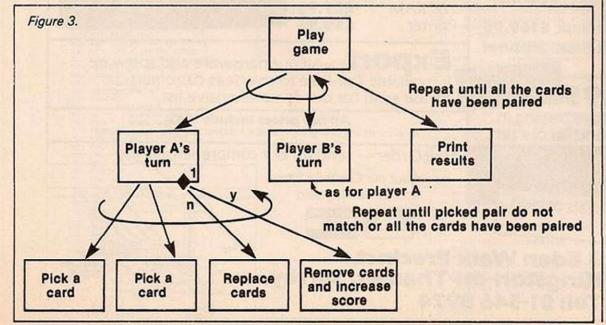
Clearly, the program will get stuck in the loop which calls procedure B for ever and a day. Of course, the real code to be placed in procedure

B will do something which results in Flag being set to "1", but you have not written this yet. So, once you have tested the program as above, simply insert a new statement 2020, setting Flag equal to "1". This will force the program to leave the loop and execute the later code.

To return to the Pelmanism problem, we are approaching an adequate definition, but need to be more precise. For example, the box "Pick a card" needs to be expanded as in figure 4.

Let us now recap on what we have been doing. First, we defined our problem at the highest meaningful level, then at the next level down. We were then able to code our program in skeleton form. As further understanding came, we could expand our structured design downwards into increasing levels of detail, and, at each level, could insert code into "black boxes". This process can be summarised as:
DEFINE OBJECTIVE
DEFINE PROBLEM ELEMENTS
AT CURRENT LEVEL
CODE SKELETON SOLUTION
IF PROGRAM COMPLETE, STOP
OTHERWISE

The more astute of you will be a little cynical about the glib way we suggest you flip from adding a box to the diagram to actually coding it — and quite right, too. The coded solution to one box may, of course, be far from simple. It may require some thought as to how you can program it. Again, we urge you not to try to develop this code at the keyboard. You will find your time is far more productive if you try to sketch out the required code beforehand, using some kind of shorthand notation. This is where we can learn from the experience of professionals, and introduce the concept of "pseudo-code".

Various academics have written doubtless worthy tomes on language systems for pseudo-code. Some advocate a free-form representation, similar to Basic, whilst others prefer to have you fill in complicated-looking forms. Our advice is to forget all that, unless you are intent on immersing your hobby in a bureaucracy. You should develop your own shorthand notation — after all, you are the only person who will have to understand it. It is different for professionals in a big department, where other programmers have to pick up the documentation and work from it.
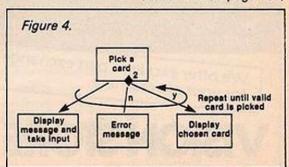
What does pseudo-code look like? It is somewhere between a precise form of English and the program code itself. Its syntax is loose, but its format should be consistent and clear. The pseudo code for the box "Set-up game" in the Pelmanism problem could be:
Proc set up game
 — Set up deck
 — Set up a string of card values

Figure 3.

Figure 4.

— Establish address of string
Shuffle cards
 — For I = 1 to 20
 — Select two card values at random
 — Exchange position of card values in string
 — Next I
 — Display cards
End Proc set-up game

You can see that pseudo-code enables you to resolve most of the problems due to logical flow, off-line. Nothing you do here can wreck the work you have already coded. Once you have fully desk-checked your pseudo-code, it is a straightforward matter to put it into the pedantic language your computer expects.

We know from experience that this methodology works — not just for us, but for thousands of people in the professional programming business as well. It produces the correct results quicker, and that, as we said earlier, does make programming considerably more fun.

Figure 5 contains the full listing of the Pelmanism program, which was developed using a structured design approach. It is written in BBC Basic.

This particular solution uses the Teletext Mode (7), but it could, with benefit, be adapted to use high-resolution graphics mode, with proper pictures of the cards.

Lines	40-110	Main body of program
		Call routines to Set-up game
		Play game and Print results
Lines	120-470	Procedure to set up game
		Card backs are represented by graphics symbols
Lines	480-620	Play Game
		Uses keys to move cursor, and select card.
		I-up; M-down; J-left; K-right; R-reveal.
Lines	630-780	Player's turn
Lines	790-870	Pair match
Lines	880-970	No match
Lines	990-1130	Scan keyboard for input
Lines	1140-1370	Move cursor
Lines	1380-1500	Turns over card
Lines	1510-1680	Calculate which card to turn over
Lines	1690-1800	Input players' names
Lines	1910-1910	Prints final scores

(Pelmanism program)

```
10 REM     **************
20 REM     * MAIN BODY *
30 REM     **************
40 MODE 7
50 CLS
60 PROCSETUP
70 PROCNAME
80 PROCGAME
90 PROCRESULT
100 *FX 4,0
110 END
120 DEF PROCSETUP
130 REM     **************
140 REM     * SET UP PACK *
150 REM     **************
160 CARD$=STRING$(4,"A23456789TJQK")
170 CARD=&3A00
180 *CARD=CARD$
190 REM     **************
200 REM     * SHUFFLE PACK *
210 REM     **************
220 FOR I=1 TO 200
230   R1=RND(51)
240   R2=RND(51)
250   TEMP=CARD?R1
260   CARD?R1=CARD?R2
270   CARD?R2=TEMP
280 NEXT I
290 REM     **************
300 REM     * DISPLAY DECK *
310 REM     **************
320 VDU 12
330 DS1$=" 9"+STRING$(6," 9")
340 DS2$="9"+STRING$(7," 9")
350 BORDER$=STRING$(31,",")
360 PRINT'CHR$(145);" ";CHR$(60);BORDER$;CHR$(108)
370 FOR I=1 TO 3
380   PRINTCHR$(145);" ";CHR$(53);SPC(31);CHR$(106)
390   PRINTCHR$(145);" ";CHR$(53);" ";DS1$;" ";CHR$(106)
400   PRINTCHR$(145);" ";CHR$(53);SPC(31);CHR$(106)
410   PRINTCHR$(145);" ";CHR$(53);" ";DS2$;" ";CHR$(106)
420 NEXT
430 PRINTCHR$(145);" ";CHR$(53);SPC(31);CHR$(106)
440 PRINTCHR$(145);" ";CHR$(53);" ";DS1$;" ";CHR$(106)
450 PRINTCHR$(145);" ";CHR$(53);SPC(31);CHR$(106)
460 PRINTCHR$(145);" ";CHR$(45);BORDER$;CHR$(46)
470 ENDPROC
480 DEF PROCGAME
490 REM     **************
500 REM     * GAME STARTS HERE *
510 REM     **************
520 SCOREA=0
530 SCOREB=0
540 REPEAT
550   SCORE=0
560   PROCTURN(PLY1$)
570   SCOREA=SCOREA+SCORE
580   SCORE=0
590   IF SCOREA+SCOREB < 26 THEN PROCTURN(PLY2$)
600   SCOREB=SCOREB+SCORE
610 UNTIL SCOREA+SCOREB > 25
620 ENDPROC
630 DEF PROCTURN(PLY$)
640 REM     **************
650 REM     * PLAYERS TURN *
660 REM     **************
670 REPEAT
680   PRINTTAB(0,21);"PICK YOUR FIRST CARD ";PLY$;SPC(10)
690   PROCMOVE
700   CD1=CARD?REV
710   AD1=ADDRESS
720   PRINTTAB(0,21);"PICK YOUR SECOND CARD ";PLY$;SPC(10)
730   PROCMOVE
740   CD2=CARD?REV
750   AD2=ADDRESS
760   IF CD1=CD2 THEN PROCPOINT ELSE PROCFINISH
770 UNTIL CD1 <> CD2 OR SCORE+SCOREA+SCOREB > 25
780 ENDPROC
790 DEF PROCPOINT
800 REM     **************
810 REM     * "PAIR" FOUND *
820 REM     **************
830 FOR I=1 TO 2500:NEXT
840 ?AD1=32
850 ?AD2=32
860 SCORE=SCORE+1
870 ENDPROC
880 DEF PROCFINISH
890 REM     **************
900 REM     * NO MATCH *
910 REM     **************
920 FOR I=1 TO 2500:NEXT
930 AD1?(-1)=145
940 ?AD1=57
950 AD2?(-1)=145
960 ?AD2=57
970 ENDPROC
980 DEF PROCMOVE
990 REM     **************
1000 REM     * MOVE "CURSOR" *
1010 REM     **************
1020 *FX 4,1
1030 PRINTTAB(4,15);
1040 FLAG=0
1050 REPEAT
1060   IN$=INKEY$(0)
1070   IF IN$="I" THEN PROCUP
1080   IF IN$="J" THEN PROCLEFT
1090   IF IN$="K" THEN PROCRIGHT
1100   IF IN$="M" THEN PROCDOWN
1110   IF IN$="R" THEN PROCREVEAL
1120 UNTIL IN$="R" AND FLAG=1
1130 ENDPROC
1140 DEF PROCUP
1150 REM     **************
1160 REM     * MOVE CURSOR UP *
1170 REM     **************
1180 IF VPOS > 3 THEN VDU 11:VDU 11 ELSE VDU 7
1190 ENDPROC
1200 DEF PROCDOWN
1210 REM     **************
1220 REM     * MOVE CURSOR DOWN *
1230 REM     **************
1240 IF VPOS < 15 THEN VDU 10:VDU 10 ELSE VDU 7
1250 ENDPROC
1260 DEF PROCLEFT
1270 REM     **************
1280 REM     * MOVE CURSOR LEFT *
1290 REM     **************
1300 IF POS > 4 THEN VDU 8:VDU 8 ELSE VDU 7
1310 ENDPROC
1320 DEF PROCRIGHT
1330 REM     **************
1340 REM     * MOVE CURSOR RIGHT *
1350 REM     **************
1360 IF POS < 32 THEN VDU 9:VDU 9 ELSE VDU 7
1370 ENDPROC
1380 DEF PROCREVEAL
1390 REM     **************
1400 REM     * REVEAL CARD *
1410 REM     **************
1420 X=POS
1430 Y=VPOS
1440 ADDRESS=HIMEM+(40*Y)+X
1450 IF ?ADDRESS <> 57 THEN ENDPROC
1460 FLAG=1
1470 IF VPOS=5 OR VPOS=9 OR VPOS=13 THEN PROCEIGHT ELSE PROCSEVEN
1480 ADDRESS?1=145:ADDRESS?(-1)=131
1490 ?ADDRESS=CARD?REV
1500 ENDPROC
1510 DEF PROCSEVEN
1520 REM     **************
1530 REM     * FIND CARD *
1540 REM     **************
1550 Y1=INT(Y/2)
1560 X1=(X-2)/4
1570 Y2=INT(Y1/2)
1580 REV=((Y2*15)+X1)-1
1590 ENDPROC
1600 DEF PROCEIGHT
1610 REM     **************
1620 REM     * FIND CARD *
1630 REM     **************
1640 Y1=INT(Y/2)
1650 X1=X/4
1660 Y2=INT(Y1/2)-1
1670 REV=(7+(Y2*15)+X1)-1
1680 ENDPROC
1690 DEF PROCNAME
1700 REM     **************
1710 REM     * INPUT NAMES *
1720 REM     **************
1730 PRINTTAB(0,21)"PLAYER 1 WHAT IS YOUR NAME "
1740 INPUT PLY1$
1750 PRINTTAB(0,22);SPC(40)
1760 PRINTTAB(0,21)"PLAYER 2 WHAT IS YOUR NAME "
1770 INPUT PLY2$
1780 PRINTTAB(0,22);SPC(40)
1790 PRINTTAB(0,21);SPC(40)
1800 ENDPROC
1810 DEF PROCRESULT
1820 REM     **************
1830 REM     * PRINT FINAL SCORE *
1840 REM     **************
1850 VDU 12
1860 PRINTTAB(5,5);PLY1$;" YOU HAVE SCORED ";SCOREA
1870 PRINTTAB(5,8);PLY2$;" YOU HAVE SCORED ";SCOREB
1880 IF SCOREA > SCOREB THEN PRINTTAB(3,13);CHR$(141);CHR$(136);PLY1$;CHR$(137);"YOU'VE WON !",TAB(3,14);CHR$(141);CHR$(136);PLY1$;CHR$(137);"YOU'VE WON !"
1890 IF SCOREB > SCOREA THEN PRINTTAB(3,13);CHR$(141);CHR$(136);PLY2$;CHR$(137);"YOU'VE WON !",TAB(3,14);CHR$(141);CHR$(136);PLY2$;CHR$(137);"YOU'VE WON !"
1900 IF SCOREA = SCOREB THEN PRINTTAB(3,13);CHR$(141);CHR$(136);"IT'S A DRAW !",TAB(3,14);CHR$(141);CHR$(136);"IT'S A DRAW !"
1910 ENDPROC
1920 DEF PROCINSTR
1930 REM     **************
1940 REM     * INSTRUCTIONS *
1950 REM     **************
1960 CLS
1970 PRINTTAB(11,3);CHR$(135);CHR$(157);CHR$(129);CHR$(141);"PELMANISM   ";CHR$(156)
1980 PRINTTAB(11,4);CHR$(135);CHR$(157);CHR$(129);CHR$(141);"PELMANISM   ";CHR$(156)
1990 PRINTTAB(0,10);
2000 INPUT"DO YOU REQUIRE INSTRUCTIONS ",ANSWER$
2010 IF MID$(ANSWER$,1,1) <> "Y" THEN ENDPROC
2020 PRINTTAB(0,8);" PELMANISM is a simple card game for"'" two players.The cards are placed face"'" down,the object is to turn up pairs"
2030 PRINTCHR$(130);CHR$(130);" Each player takes it in turn to try";CHR$(130);"and pick two cards which match,if they";CHR$(130);"are a pair then they are removd from"
2040 PRINTCHR$(130);"the pack and the successful player gets";CHR$(130);"another turn."''CHR$(129);"The winner is the player who has turned";CHR$(129);"over the.greatest number of pairs at"
2050 PRINTCHR$(129);"the end."
2060 PRINTTAB(10,24);CHR$(131);CHR$(136);"Press any key to continue"
2070 A$=GET$
2080 CLS
2090 PRINTTAB(11,3);CHR$(135);CHR$(157);CHR$(130);CHR$(141);" CONTROLS   ";CHR$(156)
2100 PRINTTAB(11,4);CHR$(135);CHR$(157);CHR$(130);CHR$(141);" CONTROLS   ";CHR$(156)
2110 PRINTTAB(0,10);CHR$(129);" I ";CHR$(135);"- TO GO UP"
2120 PRINTTAB(0,12);CHR$(129);" M ";CHR$(135);"- TO GO DOWN"
2130 PRINTTAB(0,14);CHR$(129);" J ";CHR$(135);"- TO GO RIGHT"
2140 PRINTTAB(0,16);CHR$(129);" K ";CHR$(135);"- TO GO LEFT"
2150 PRINTTAB(0,18);CHR$(129);" R ";CHR$(135);"- TO REVEAL THE CARD"
2160 PRINTTAB(10,24);CHR$(131);CHR$(136);"Press any key to continue"
2170 A$=GET$
2180 ENDPROC
```

MAKING USE OF SPECTRUM INTERRUPTS

Everything from software clocks to automatic screen scrolling — Nigel Dore shows you how to divert the interrupt to user routines to create a very powerful facility

AN INTERRUPT is a signal sent to the microprocessor which generally suspends the execution of the current program — without the program knowing it.

Every 20 ms the Z-80 processor in the Spectrum receives an interrupt which forces it to suspend what it is currently doing and to jump to 38 hexadecimal or 56 decimal where it increments the system variable frames and scans the keyboard so that the Spectrum knows what you are keying in, if anything. If this interrupt could be diverted elsewhere to a user routine it would provide a very powerful facility which would have almost limitless uses. The way in which the Z-80 responds to an interrupt is governed by which interrupt mode it is operating in; the Z-80 has three modes.

Interrupt mode 0 allows the interrupting device to place one instruction on to the data-bus after the interrupt for execution by the Z-80. Interrupt mode 1 makes a jump to 38hexadecimal or 56 decimal and this is the mode that the Spectrum uses. Interrupt mode 2 is more complex. When the Z-80 receives an interrupt, it expects the interrupting device to place one byte of data on the data bus which acts as the low order of an address. The high order of the address comes from the I register. The Z-80 looks at this address which gives it a second address that it then jumps to.

The last mode provides a method of diverting the Z-80 to a user routine for, although the Spectrum does not provide the byte of data required, this means the Z-80

16K mother program.

```
   10 REM Spectrum Interupts
   20 REM for 16K Spectrum
   30 REM @N.DORE
   35 CLEAR 32329: LET c=0
   40 FOR i=32330 TO 1e9: READ a:
IF a<256 THEN LET c=c+a: POKE i
,ABS a: NEXT i
   70 IF a<>c THEN PRINT "ERROR I
N PROGRAM": STOP
  100 REM ON/OFF routines
  110 DATA 62,40,237,71,237,94,20
1,0,0,0,62,62,237,86,237,71,201,
0
  120 REM SERVICE ROUTINE
  130 DATA 255,243,197,213,229,24
5,1,254,127,237,120,254,252,40,9
,205,144,126,241,225,209,193,251
,201,62,56,50,141,92,205,107,13,
1,254,127,237,120,254,252,40,247
,251,207,20,0,0,0,0,0,0,0,0
  140 REM User routine(here RET)
  150 DATA 201,9306
```

16K mother program Z-80 code.

```
ON ROUTINE
ld a,40                    62  40
ld i,a                     237 71
im 2                       237 94
ret                        201
OFF ROUTINE
ld a,62                    62  62
im 1                       237 86
ld i,a                     237 71
ret                        201
SERVICE ROUTINE
rst 56                     255
di                         243
push bc                    197
push de                    213
push hl                    229
push af                    245
ld bc,32766                1 254 127
in a,(c)                   237 120
cp 252                     254 252
jr z,+9                    40  9
call 32400                 205 107 13
pop af                     241
pop hl                     225
pop de                     209
pop bc                     193
ei                         251
ret                        201
ld a,56                    62  56
ld (23693),a               50 141 92
call cls(3435)             205 107 13
ld bc,32766                1 254 127
in a,(c)                   237 120
cp 252                     254 252
jr z,-9                    40 247
ei                         251
rst 8(reports)             207
error break                20
```

Figure 5 This generates a moving landscape at the bottom of the screen which can be used in one of your own games. The speed at which the ground moves can be changed by Poking 23681 which should be followed by Poke 23672,0. This program should be saved before being run.

```
  140 REM moving land data
  150 DATA 58,120,92,33,129,92,70
,184,216,62,0,50,120,92,33,95,90
,6,160,126,254,0,32,7,54,56,43,5
```

thinks it sees FFh or 255 decimal. Thus we need only change that I register so that the address

$$I*256 + 255$$

and the following one point to our user routine. However, the I register cannot be moved so that

$$I*256$$

is in RAM without causing screen interference.

Therefore we must find our vector address somewhere in ROM. A quick search finds suitable vectors for both 16K and 48K machines:

$$I = 40$$

will produce a jump to 32348 for 16K machines,

$$I = 9$$

will produce a jump to 65129 for 48K machines. In order to see what can be done with these interrupts first type in the mother program for your particular machine which allows all the following programs to be compatible between both machines. After typing in the program, Run it. If after a few seconds, an "Error in program" message appears, then carefully check the Data statements and try again. If "OK" appears then save the program as the checksum does not detect all possible errors. Here are the commands for turning the routine on or off:

```
ON
16K LET a=USR 32330
48K LET a=USR 65180
OFF
16K LET a=USR 32340
48K LET a=USR 65190
```

Turn the routine on. If the computer crashes, then re-load the program and check it carefully. If "OK" appears on the screen then the routine should be working. The mother program on its own adds one useful feature: the depression of symbol-shift and space together now acts as Break but with these added advantages. The new Break works in machine-code loops, it clears the screen and returns the colours to the normal black on white. The use of this break when not running a program causes a buzz with Colour Reset and Clear Screen.

To see what the routine can do, turn it off and add the lines in figure 1 erasing the original lines of the same line number. Make sure you have turned the routine off and Run the new program. Turn the routine back on and you will be greeted by a pulsating border display and a clicking sound which continues while your own programs are running.

You might like to try loading a Basic program off a cassette. The border display will stop during the load, as it will during Beep, for the computer disables the interrupts because timing is important. The border will start pulsating again as soon as the program has loaded. This program achieves its effect by outputting the contents of the frame counter to the port which controls the border 50 times a second.

The other five examples can be entered in the same way but remember to take great care that the old routine is turned off before running the mother program to enter the new routine. ■

```
4,0,35,0,35,16,241,42,121,92,35,
34,121,92,38,1,70,56,176,92,254,
1,40,23,58,177,92,203,64,40,4,60
,50,177,92,254,5,32,29,62,1,50,1
76,92,24,22,0,56,177,92,203,64,4
0,4,61
 160 DATA 50,177,92,254,1,32,6,6
2,2,50,176,92,0,56,177,92,71,17,
32,0,33,255,90,54,0,183,237,82,1
6,249,71,62,6,144,71,54,56,183,2
37,82,16,249,201,19586
 200 POKE 23728,1: POKE 23729,2:
POKE 23681,1:
```

48K Mother program

```
 10 REM Spectrum Interupts
 20 REM for 48k Spectrum
 30 REM @N.DORE
 35 CLEAR 64911. LET c=0
 40 FOR i=65129 TO 1e9: READ a:
IF a=256 THEN LET i=64912: NEXT
i
 50 IF a<256 THEN LET c=c+a: PO
KE i,ABS a+(a<0)*127: NEXT i
 70 IF a+96<>c THEN PRINT "ERRO
R IN PROGRAM": STOP
 100 REM SERVICE ROUTINE
 110 DATA 255,243,197,213,229,24
5,1,254,127,237,120,254,252,40,9
,205,144,253,241,225,209,195,251
,201,62,56,50,141,92,205,107,13
1,254,127,237,120,254,252,40,247
,251,207,20,0,0,0,0,0,0,0
 120 REM ON/OFF routines
 130 DATA 62,9,237,71,237,94,201
,0,0,0,62,62,237,66,237,71,201,5
56
 140 REM User routine(here RET)
 150 DATA 201,9306
```

Figure 1.

```
 140>REM border display
 150 DATA 58,120,92,211,254,201,
10041
```

Figure 2 This shows you how your Spectrum can generate sounds while you enter or run a program.

```
 140>REM sound
 150 DATA 42,120,92,203,61,38,0,
41,41,41,41,17,2,0,205,181,3,201
,10434
```

Figure 3 This makes the screen scroll automatically unless the n key is pressed.

```
 140>REM Non-stop scroll
 150 DATA 1,254,127,237,120,254,
247,200,62,5,50,140,92,201,11096
```

Figure 4 This provides an On Error Beep function. It causes the computer to beep when an error occurs until Enter is pressed.

```
 140>REM On error bleep
 150 DATA 58,58,92,254,255,200,1
7,150,0,33,100,0,205,181,3,243,2
01,11155
```

Figure 6 This last routine generates a display of the 24-hour clock at the top right of the screen. The computer will even beep on the hour, all while your own program runs. Save this program before running it.

```
 140>REM clock data
 150 DATA 58,73,-127,50,121,92,5
5,120,92,22,0,32,78,50,4,163,140
,92,35,82,-127,14,0,64,1,43,126,
254,32,40,250,50,12,119,62,3,16,
40,130,24,7,0
 160 DATA 126,254,7,40,222,0,33,
72,-127,126,63,32,14,35,126,
54,-32,43,43,43,43,11,16,16,251,
58,121,92,87,56,73,-127,136,196,
55,-127
 170 DATA 17,21,54,33,71,-127,35
,229,213,126,71,254,32,40,33,35
,126,50,120,81,25,18,253,221,35
,6,8,126,235,119,235,20,35,16,24
,6,126,254,255,119,235,20,35,16,24
 175 DATA 33,21,58,6,10,54,120,
```

```
5,16,251,62,127,50,20,86,50,23,6
8,58,89,50,89,31,88,50,205,85,201
,8,0,0,17,128,9,33,0,1,205,181,3
,243,201,22160
 176 REM 
 180 DIM a$(10): INPUT "Hour ";h
"Minutes ";a: LET a$="0" AND
LEN STR$ a=1+STR$ a+"0"+(10" AND
LEN STR$ h=1)+STR$ h+"0"+(10" AND
FOR i=1 TO 10: POKE i*325634(PEEK
23733/130)*32812,CODE a$(i)-47:
NEXT i
 190 INPUT "Key to start";a$: PO
KE (130)*32330+(PEEK 23
33/130)*32330+(PEEK 23733/130)*6
5180)
```

VOICES OF THE VIC

While the Vic-20's four-channel sound facility is one of its most appealing features, do you really know how to make the most of them? John Ransley explores the electronic quartet's potential.

YOUR UNEXPANDED Vic-20 can make you feel like Richard Clayderman, Acker Bilk, John Williams or James Galway. But the big question often asked by Vic owners is, how? After all, my copy of Commodore's introductory manual manages to transpose whole chunks of instructions in that section briefly describing the micro's music-making capabilities — so anyone else attempting to exploit them might be forgiven for taking a powder and going for a long lie-down in a quiet darkened room.

So start again — from here, with some easy routines which I will explain as we go along. Incidentally, if your micro is not a Vic-20, you will find that it is still possible to foster many of these programs, usually by simply substituting for any Poke statements the equivalent Basic command in your micro's dialect, which will probably be something like Sound or Beep.

You can make creative use of at least three of the Vic's four sound generators, individually or collectively, and in several different ways. First, you can use them to play the Vic's keyboard like that of a piano or organ; secondly, the Vic can be a musician, playing itself the music you give it to learn, just as though it was an electronic pianola; thirdly, it can help you compose your own music by combining the two facilities just mentioned; fourth, the Vic can take over the actual composing and produce sonorous fugues or perky rags — and in all the programs that follow, I have incorporated some kind of screen display to help you

understand even better what's going on in the Vic's mind.

The Vic's soundmakers slumber silently in four separate Poke locations: 36874, 36875, 36876 and 36877. The first three of these are always used, alone or collectively, to play music. The fourth, 36877, is the white noise generator — all of whose tones sound like nothing so interesting as an untuned radio until you start shaping them.

Of the three music voices, it is convenient to think of 36874 as being the lowest or bass voice; 36875 as the mid-range or tenor voice; and 36876 as the highest, or soprano. Each of these three voices has a library of 126 tones numbered 128 to 254 inclusive, covering some three octaves of the conventional music scale. Poke 36874 drops an octave lower than, and Poke 36876 an octave higher than, Poke 36875. Therefore, you have something like a five-octave range across the three voices.

The only other Poke location you will need is 36878, which is qualified by a number from

0 to 15. This command switches on the volume control and sets the sound level. You may find that the highest setting distorts when played through the speaker of your particular TV or monitor; Poke 36878,10 or Poke 36878,12 should be about right most times.

Type New, hit Return, and enter Program 1. Run this, and you hear each of the 127 tones held in the Vic's bass voice, and see its number displayed on-screen. Everything stops at 254 except the noise. Hit Run, Stop and Restore, then List Program 1. Tone 254 continued because I did not include a line cancelling the volume. So add these lines:

```
170 POKE 36878,0
180 POKE 36874,0
```

It is not strictly necessary to include line 180 in this program, but if you do not the old voice will stay live and still be heard behind any other voice Poked.

Each of the three Poke locations is capable of generating 126 notes. Change line 130 in Program 1 to Poke 36875,T and Run; and then to Poke 36876,T and Run to hear the tenor and soprano equivalents respectively. With these, the two or three uppermost tones are at such a high frequency that you may not even be able to hear them.

At this stage, it is important to draw a distinction between tones and notes, because only 25 of the tones in each Poke location are more or less equivalent to the notes ABCDEFG of the conventional musical scale. And to complicate matters, they match up in uneven steps. For example, note E is tone 206, F is 209 and F# is 211. The whole range of 126 available tones thereefore gives you just three octaves, from a low C on tone 131 to a high C on tone 224.

This next simple program proves the Vic's ability to produce an orderly musical scale of eight notes — C major. New the memory and enter Program 2. In this program, lines 110 to 130 set up a store cupboard of tones equivalent to the scale of C major and their names; count up the pairs and you will find they total eight in number. Lines 140 and 160 form a loop that makes line 150 read through each Data pair and store them in memory as matched sets tagged 1 to 8.

At line 170, the Vic is waiting for you to tell it which pair you want brought up front, by touching any one of the keys numbered 1 to 8 on the top row of the keyboard; the second half of line 170 makes the Vic go round in circles until you do so.

Line 180 takes the value of the key you have touched and marries it up with the Data pair tagged with that same number. So if you hit key 4, the fourth pair along in the Data queue — which happens to be 232,F — gets yanked out and put to work. On goes the volume in line 190, and line 200 selects the tenor voice and drops in the (N)ote line 180 produced. The other half of the pair, the actual letter C,

then neatly drops into the Print statement in 210. So as you touch each key, you will hear a note in C major ranging from the lowest (triggered by key 1) to the highest (key 8), and receive visual confirmation of the note the Vic played. If you want to temporarily mute the sound, touch the space bar — or indeed, any key other than 1 to 8.

You can give yourself a little more self-tuition in equating the Vic's tone numbers with actual note values by expanding a little on the Data items. Just as an experiment, retype line 110 of Program 2 as follows:
110 DATA 225,C (225), 228,D (228), 231,E (231)

Run this revised version and when you hit key 1, 2 or 3 the screen will display not only the note name as before but also, in brackets, the tone number used to produce that note. This can be a useful aid when you are compiling Data statements from sheet music.

Before moving on to programmed music, using Data statements as scores on which the Vic can work, let us try brightening up the general sound produced by Program 2. Program 3 will teach you a new technique for shaping the character of the Vic's voices, as well as producing a new tune.

As you are now entering the realm of longer and more complicated Data queues, perhaps it is worth mentioning the most common reasons why the Vic flashes up the dreaded Out of Data error message. One is that you have missed out a comma somewhere; there must be one after every Data item except the last in any one Data line. Another is that you have typed a letter O instead of the special slashed zero, 0. Out of Data will also show up if the number given in the Read loop does not match the number of items in Data.

Finally, if you are adapting any of these routines to use in a more ambitious program, remember that you may have to set up a Dim statement to reserve memory for the Read array. If you do not, you may prompt the Bad Subscript error message.

Back to the music. Again, New the memory, and carefully type in Program 3. This sounds a little more like a piano, thanks to lines 170 to 200. These ensure that the volume of each note is heard at full volume and then dies away to nothing after the time allowed to play the (L)ength of the note in line 190. If you would like to prove to yourself that this is really happening, just add this one line to the program:
205 PRINT " [CLR SCR] " N; D; L

If all is well, the screen will display a different tone number for each note that is played, then the digit 0 and the number 31. Check through the logic of Program 3 and you should be able to understand why.
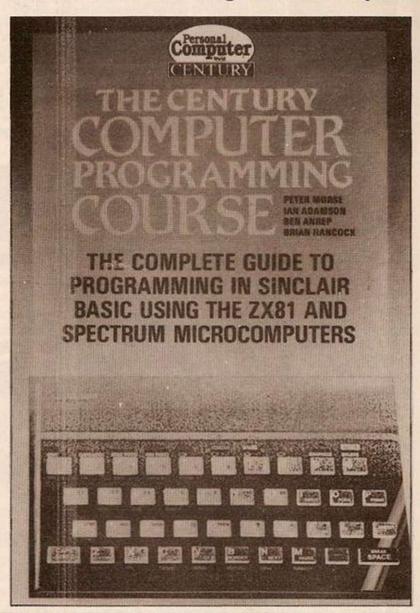
This exercise also introduces a couple of other little tricks. First, perhaps you noticed the dummy value of 0 at the end of line 120. That is because, unlike Program 2, the Read statement in line 140 simply starts at the first item of Data and doggedly works its way right through the queue, item by item. The 0 signals to the second part of line 140 that the notes are all used up and so the tune is over. Without it, the program would jam on the last note and flip up the news Out of Data Error in

Program 1.
```
100 REM PROGRAM 1
110 POKE 36878,10
120 FOR T=128 TO 254
130 POKE 36874,T
140 PRINT T,
150 FOR D=1 TO 400
160 NEXT D:NEXT T
```

Program 2.
```
100 REM PROGRAM 2
110 DATA 225,C,228,D,
    231,E
120 DATA 232,F,235,G,
    237,A
130 DATA 239,B,240,C
140 FOR N=1 TO 8
150 READ S(N), N$(N)
160 NEXT N
170 GET K$:IF K$="""
    THEN 170
180 N=VAL(K$)
190 POKE 36878,10
200 POKE 36875, S(N)
210 PRINT"▓▓▓▓▓▓▓▓▓▓▓
    YOU HIT NOTE ";N$(N)
220 GOTO 170
```

Program 7.
```
100 REM PROGRAM 7
110 N=INT(RND(1)
    *30)+195
120 POKE 36878,10
130 POKE 36875,N
140 PRINT"I'M PLAYING
    TONE";N
150 FOR D=1TO400:
    NEXT D
160 POKE 36878,0
170 GOTO 110
```

Program 4.
```
100 REM PROGRAM 4
110 DATA 215,400,217,
    600,215,400,209,
    600,207,400,209,
    600
120 DATA 215,400,221,
    600,217,400,215,
    600,209,400,217,
    600
130 RESTORE
140 READ N:IF N=0
    THEN220,0
150 READ D
160 POKE 36875,N
170 POKE 36878,10
180 FOR L=1 TO
    D:NEXT L
190 POKE 36878,0
200 GOTO 140
210 POKE 36874,
    0:POKE36878,0
220 END
```

Program 6.
```
100 REM PROGRAM 6
110 N=INT(RND(1)
    *129)+126
120 POKE 36878,10
130 POKE 36875,N
140 PRINT"I'M PLAYING
    TONE";N
150 FOR D=1TO400:
    NEXT D
160 POKE 36878,0
170 GOTO 110
```

Program 5.
```
100 REM PROGRAM 5
110 DATA 201,800,209,400,225,1200,221,
    800,209,400,203,1200
120 DATA 201,800,201,400,201,400,203,
    400,209,400,215,1200,209,1200
130 DATA 201,800,209,400,225,1200,221
    ,800,209,400,203,1200
140 DATA 201,800,209,400,209,400,215,
    400,219,400,221,1200,221,1200,5
150 RESTORE
160 READ N:IF N=5 THEN 220
170 READ V
180 POKE 36878,10:POKE36876,N
190 FOR D=1 TO V:NEXTD
210 GOTO 160
220 POKE 36876,0:POKE 36878,0
230 END
```

140. If you like, try this for yourself by cursoring out the dummy zero and then Running the program.

Secondly, the Restore statement in line 130 is not strictly necessary in this particular program as you have to rerun it every time you want to hear the music anyway. But usually, you will want to set Read's pointer to the head of the Data queue in line 110 in case you want to call it again; Restore does just that.

Before we go on to refine Program 3 by giving each of the notes proper values, do not overlook the creativity which lines 170 to 200 offer you. You can tinker with these along with lines 150 and 160 to get a sound more like a bar-room piano than a Bechstein, for example. Lower the 15 to 0 range of line 170 and you will soften the apparent attack; alter the Step value in the same line — even to a decimal such as .25 — and you can remould the sustain and rate of decay.

You could try adding 36876 as well, or trying different combinations of the three. Further, changing the value of the delay loop in line 190 alters the overall speed at which all this happens. This little suite of three variables is one of the most powerful tools you have in exploiting the Vic's soundmaking potential.

So Program 3 opened some new horizons, but to hear the music played properly, we need to pair every note in the Data queue with a value representing its proper duration, just as we added the letters of the scale in Program 2.

Tidy the Vic's memory by typing New and hitting Return, and this time we will have James Galway audition for Dooley Wilson's job at the Café Américain in program 4.

The deliberate error is at line 190. That Poke is chopping up the notes in a way that a flautist certainly does not do. So delete line 190 altogether and Run the program again. Now the notes do glide into one another as they should.

In this exercise I have used a value of 3,000 to the bar. It is an easy matter to quicken or slow the overall tempo at which this or any other tune is played, by simply changing proportionally the value given to each D item in Data. Thus you could accelerate Mr Galway by changing 400s to 300s, and the 600s to 450s, and so on.

But beware, this deceptively easy program has a trap for the unwary. New memory and enter Program 5.

When you Run this program, all will be well until you hear the Vic play those notes in lines 120 and 140 which are the same in tone, if not duration, of those preceding them. Just drop in a pair of zeros to achieve an imperceptible break. You may use this same trick to imitate the rests shown on sheet music. You can specify the length of the rest by giving N a proper value to work on. As an example:

    DATA 201,400,0,200,201,400

will sound tone 201 for, say, a half-note; then introduce a quarter-note rest; and then play 201 for another half-note. You may also have noticed that the end of data signals at the end of lines 140 and 160 were changed to 5 from the usual 0, so that when you correct this program by adding double zeros to lines 120 and 140, your program will not end pre-

maturely. You can go on qualifying Data lines with yet more items, provided that they are always in logical sets.

You might, for example, have the notes and durations for a melody line as elements one and two in a four item set; with a bass line and associated note durations or rests as the third and fourth elements. This way you can achieve a very effective syncopated or walking bass rhythm. For a demonstration of this technique, I can do no better than refer you to Adam Macielinski's ingenious boogie programs in the March 1983 issue of *Your Computer*. Adding the Vic's third voice will give you three-part harmony in a carefully structured Data queue comprising six-element items; and maybe now is the time to experiment more with Poke 36877 to drop in a snare drum.

After all that your fingers could do with a rest, so why not let the Vic get on with some composing of its own. Program 6 generates a random number between 128 and 254, which becomes the tone qualifier for line 130; and line 140 prints it out for you. The tone plays for a time set by the loop in line 150 and the program then loops back for its next randomiser to give a continuous demonstration of *musique silicon* if not *concrète*.

The snag is that line 110 draws from too wide a choice of tones, many of which do not equate to real musical notes anyway. You can improve matters a little by reducing the choice for N to, say, just one middle octave. Make this alteration:

    100 REM PROGRAM 7
    110 N + INT(RND(1)*30) + 195.

Better, but you are still hearing some bum notes that make it sound rather strange. The best solution comes with our old friend the Data statement. New memory and enter Program 8.
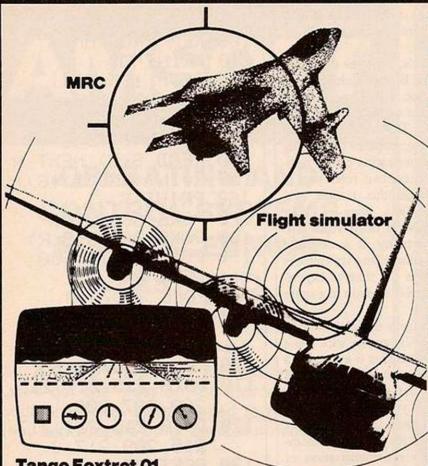
Here, lines 110, 120 and 130 set up a single octave of C major while 150 generates at random a number between 1 and 8; line 160 puts this on-screen. The volume is set in line 170, and 180 delves into the Data queue at that position fixed by R. Line 200 gives a moment of silence before looping back for the next cycle.

This program will generate music that sounds more "true" — now it is just plain monotonous thanks to line 190 sounding each note for precisely the same length of time. But it is an easy matter to introduce a further randomiser which will select different note values — the length and relationship of which you can fix. Revise the last program by changing it as follows from line 190 onwards — see Program 9.

Not stunningly elegant but it does the job. The values at the end of lines 210 to 240 you can tinker with yourself; the lower the numbers, the faster the music.

There is very much more to music making and computerised composing on your Vic 20 — not to mention the sound effects capabilities which I have not got around to examining here in any detail. Still, I hope you have fun trying, testing and adapting the programs given here and that they will help you to develop the know-how and curiosity that can result in some really original efforts of your own — ones worth listening to, as well!

```
Program 8.
100 REM PROGRAM 8
110 PRINT"⬛"
120 DATA 195,201,
    207,209,215,
    219,223,225
130 FOR N=1 TO 8
140 READ S(N):NEXT N
150 R=INT(RND(1)*8)+1
160 PRINT R;
170 POKE 36878,10
180 POKE 36875,S(R)
190 FOR D=1 TO 600:
    NEXT D
200 POKE 36878,0
210 GOTO 150
```

```
Program 3.
100 REM PROGRAM 3
110 DATA 215,217,215,
    209,207,209
120 DATA 215,221,217,
    215,209,217,0
130 RESTORE
140 READ N:IF N=0
    THEN220
150 POKE 36874,N
160 POKE 36875,N
170 FOR D=15 TO 1
    STEP -1
180 POKE 36878,D
190 FOR L=1 TO 30
200 NEXT L:NEXT D
210 GOTO 140
220 POKE 36874,0:POKE
    36875,0:POKE
    36878,0
230 END
```

```
Program 9.
100 REM PROGRAM 9
110 PRINT"⬛"
120 DATA 195,201,207,
    209,215,219,223,225
130 FOR N=1 TO 8
140 READ S(N):NEXT N
150 R=INT(RND(1)*8)+1
160 PRINT R;
170 POKE 36878,10
180 POKE 36875,S(R)
190 V=INT(RND(1)*4)+1
200 PRINT V;
210 IF V=1 THEN L=150
220 IF V=2 THEN L=250
230 IF V=3 THEN L=350
240 IF V=4 THEN L=450
250 PRINT D;
260 POKE 36878,10
270 FOR D=1 TO L:NEXT
280 POKE 36878,0
290 GOTO 150
```

FINE SCROLLING
ATARI

Flashing junk, nasty jerks: two of the problems that arise when using fine scrolling in Basic. Chris Lam looks at ways in which Atari users can circumvent unpleasant experiences.

WHAT IS THE difference between fine scrolling and coarse scrolling?

Coarse scrolling is where one character at a time moves. This gives a very jerky effect. Type in program 1 and Run it. The program manipulates LMS operands to implement the coarse scroll. Any method can be used: it is just that LMS manipulation is easier. The program tells Antic — the graphics processor — where to start displaying data from. It runs through addresses 0 to 65535. We will call the addresses which are Poked to tell Antic where to start displaying screen data, DList and DList+1. Try steps of 19, 20 and 21 in line 30. Change lines 20 and 30 so that they run from 255 to 0.

Just for those who do not know what LMS operands are, LMS — load memory scan — is an instruction followed by a 16-bit address. The address of the low byte is

PEEK (560) + 256*PEEK(561) + 4

Now for fine scrolling. Add program 2 to program 1. There are two built-in scroll registers HScrol — 54276 — and VScrol — 54277 — which hold the number of pixels to be scrolled. The direction of the scroll depends on the sequence of numbers we Poke into it. Before these registers can be deployed, we have to enable the lines on the screen for fine scrolling. Lines 12 and 13 do this. In line 12, the 64 is the LMS instruction, 16 is the horizontal fine scroll enabler — 32 is the

vertical scroll enabler — and 7 is the Antic number for graphics 2. Number 6 is for graphics 1, and 2 is for graphics 0.

The registers can only be used to scroll up to 15 pixels either way. Therefore we must combine fine scrolling with coarse scrolling.

Line 58 is the loop which determines the direction and the extent of scroll. A "0 to limit" gives an upwards or left-to-right scroll while a "limit to 0 step–1" gives a downward or right to left scroll. Look at figure 1 to determine the value of limit.

Line 62 resets the screen to its starting position before the coarse scroll. The value to be Poked is always the first number in line 58.

```
Program 1.

5 GRAPHICS 2+16
10 DLIST=PEEK(560)+256*PEEK(561)
11 DLIST=DLIST+4
20 FOR N=0 TO 255
30 FOR M=0 TO 255
40 POKE DLIST,M
50 POKE DLIST+1,N
60 FOR DELAY=1 TO 100:NEXT DELAY
70 NEXT M:NEXT N
80 END

Program 2.

5 GRAPHICS 2+16
10 DLIST=PEEK(560)+256*PEEK(561)
11 DLIST=DLIST+4
12 POKE DLIST-1,64+16+7:REM EXTRA LINE
13 FOR N=2 TO 13:POKE DLIST+N,16+7:NEXT
   N:REM EXTRA LINE
20 FOR N=0 TO 255
30 FOR M=0 TO 255
```

```
40 POKE DLIST,M
50 POKE DLIST+1,N
58 FOR J=7 TO 0 STEP -1:REM EXTRA LINE
59 POKE 54276,J:REM EXTRA LINE
60 FOR DELAY=1 TO 100:NEXT DELAY
61 NEXT J:REM EXTRA LINE
62 POKE 54276,0:REM EXTRA LINE
70 NEXT M:NEXT N
80 END

Program 3.

5 GRAPHICS 2+16
6 D=PEEK(560)+PEEK(561)*256
10 GOSUB 100
11 ACT=USR(ADR(A$))
12 POKE D+3,64+32+16+7
13 FOR N=2 TO 13:POKE D+4+N,55:NEXT N
14 POKE D+18,65
15 POKE D+19,PEEK(560)
16 POKE D+20,PEEK(561)
18 POKE 1536,15
```

(listing continued on next page)

Before going on, see if you can convert the program to do a vertical fine scroll downwards.

Two problems arise with using fine scrolling on Basic. First is the flashing junk. Second is the unpleasant jerks. Both can be cured by changing scroll registers and LMS operands during the time which the screen is blank.

Type in program 3. The routine is enclosed in lines 101 onwards. This routine is executed fifty times a second exactly. You have to feed the routine with information.

POKE 203, number of pixels to scroll horizontally.
POKE 204, number of pixels to scroll vertically.
POKE 205, low byte of new start of screen RAM.
POKE 206, high byte of new start of screen RAM.

POKE 207, low byte of DLIST.
POKE 208, high byte of DLIST.
POKE 209, if Poked with a non-zero number it will load the LMS operands with what is in location 205, 206.
POKE 1536, number you would normally have on line 73 of program 2. This is the horizontal one.
POKE 1537, number you would normally have in line 73 of program 2. This is the vertical one.

This routine caters for horizontal and vertical scrolling. Program 3 demonstrates diagonal scrolling. Use this routine in your programs. You are probably saying to yourself "Who wants to watch a load of scrolling junk?". If you set up a string and manipulate LMS operands to point to the start of the string, then changing the contents of the string, we can implement coarse scroll. Try program 4.

The last program gives the skeleton on which the scrolling map in Eastern Front 1941 is created. Use your joystick to scroll the

screen over nine screens full of random characters. A final warning. The computer has an internal character set which is not the same as the standard one. Such maps require a lot of careful planning before they can be created. This is all possible because screen data is not fixed to one section of memory but can be anywhere. Imagine how difficult it would be if you had to move all the data through screen ram. Program 5 uses twelve LMS instructions which is the proper set up for horizontal scrolling.

Why not try replacing MAP$ with your own graphics? ∎

Figure 1.

Graphics mode	0	1	2	3	4	5
Horizontal 1 unit	3	7	7	3	1	1
Vertical 1 unit	7	7	15	7	3	3

```
19 POKE 1537,15
20 FOR I=255 TO 0 STEP -1
40 FOR J=255 TO 0 STEP -22
42 POKE 205,J:POKE 206,I:POKE 209,1
50 FOR N=15 TO 0 STEP -1
57 POKE 204,N
58 POKE 203,N
59 FOR K=0 TO 20:NEXT K
60 NEXT N
70 NEXT J
80 NEXT I
90 STOP
100 REM DATA
101 DATA 165,209,240,25,160,0,165,205,145,207,
    200,165,206,145,207,173,0,6,133,203
102 DATA 173,1,6,133,204,169,0,133,209,165,203,
    141,4,212,165,204,141,5,212
109 DATA 76,98,228
110 FOR N=0 TO 41:READ A:POKE 1538+N,A:NEXT N
120 DATA 160,2,162,6,169,7,32,92,228,104,96
130 DIM A$(11)
140 FOR N=1 TO 11:READ A:A$(N,N)=CHR$(A):NEXT N
150 CO=D+4
160 POKE 208, INT(CO/256)
165 POKE 207,CO- INT(CO/256)*256
170 RETURN
```

Program 4

```
5 GRAPHICS 2+16
10 DIM SCR$(240),LINE$(7)
15 SCR$(1)=" ":SCR$(240)=" ":SCR$(2)=SCR$
16 REM IN 15 PUT CTRL J BETWEEN QUOTES
20 LINE$="       "
25 REM 20 IS CTRL J,,,,,J
30 START=ADR(SCR$)
31 DLIST=PEEK(560)+256*PEEK(561)+4
32 HI=INT(START/256)
33 POKE DLIST+1,HI
34 POKE DLIST,START-HI*256
40 SVE=IN
50 RAND=INT(RND(0)*2+1)
60 IN=IN+(RAND=1)-(RAND=2)
70 IF IN=-1 OR IN=14 THEN IN=SVE
75 SCR$(1,220)=SCR$(21,240)
80 SCR$(221+IN,221+IN+6)=LINE$
```

```
90 FOR N=1 TO 50:NEXT N
110 GOTO 40
```

Program 5.

```
3 GRAPHICS 0:POSITION 15,12:?
  "PLEASE WAIT"
4 GOTO 10
5 FOR F=1 TO 60:B$(F,F)=CHR$
  (PEEK(53770)):NEXT F:RETURN
10 DIM MAP$(2160),B$(60)
20 FOR N=1 TO 2101 STEP 60
40 GOSUB 5:MAP$(N,N+59)=B$
70 NEXT N
90 BEGIN=ADR(MAP$)
100 GRAPHICS 2+16:ST=1536
120 FOR N=0 TO 2:POKE ST+N,112:NEXT N
130 FOR I=1 TO 12
140 POKE ST+3*I,71
150 MEMST=BEGIN+(I-1)*60
155 HI=INT(MEMST/256)
160 POKE ST+3*I+2,HI
170 POKE ST+3*I+1,MEMST-HI*256
180 NEXT I
190 POKE 1575,65
200 POKE 1576,0
210 POKE 1577,6
220 POKE 560,0
230 POKE 561,6
1000 X=0:Y=0
1005 POKE 712,10
1010 ST=STICK(0)
1020 IF ST=15 THEN 1010
1030 X=X+(ST=7 AND X<40)-
     (ST=11 AND X>0)
1040 Y=Y-(ST=14 AND Y>0)+
     (ST=13 AND Y<24)
1050 TMP=BEGIN+X+Y*60
1060 FOR N=1540 TO 1573 STEP 3
1070 HI=INT(TMP/256)
1075 POKE N+1,HI
1080 POKE N,TMP-HI*256
1090 TMP=TMP+60
1100 NEXT N
1110 GOTO 1010
```

MAPLIN

HARDWARE

Atari 400 with 16K RAM (AF36P) £199.95
Atari 400 with 48K RAM (AF37S) £299.00 ★
Atari 800 with 48K RAM (AF02C) £399.00 ★
All above with BASIC and handbooks
Atari Cassette Recorder (AF28F) £50.00
Atari Disk Drive (AF06G) £299.95
48K RAM for Atari 400 (AF44X) £99.95
48K Upgrade for 400 (AF45Y) £99.95
Commodore 64 (AF56L) £339.00 ★
VIC20 (AF47B) £129.95
16K RAM for VIC20 (AF53H) £59.95
Commodore Cassette (AF48C) £44.95
Commodore Disk Drive (AF50E) £299.99
Dragon 32 (AF57M) £199.50
Dragon Joystick (BC30H) £19.95
Cassette Cable for Dragon (BC31J) £2.95
Epson Printer MX80F/T (AF40T) £447.35
MENTA (XG28F) £115.00
Floppy Disk (YX87U) £2.50
5 C12 Data Cassettes (AF61R) £1.99
Joystick Controller (AC53H) £7.50
Joysticks (Pair) (AC37S) £13.95
Le Stick (AC45Y) £24.95
Full details all above in MAPLIN catalogue.

★ Delivery next day by Datapost

SELECTED SOFTWARE

ATARI

Attack At Ep-Cyg-4	-1E-16K-KF54J	£34.95
*Baja Buggies	-1C-16K-KB74R	£22.95
Candy Factory	-1D-32K-KF53H	£21.95
Claim Jumper	-1E-116K-KB67X	£34.95
Defender	-1E-16K-KF10L	£29.95
Forth	-1D-24K-YL29G	£62.95
*Frogger	-1C-16K-KB68Y	£22.95
Galaxian	-1E-16K-KF11M	£29.95
Gorf	-1E-16K-KB44X	£34.95
*Hellfire Warrior	-1C-32K-KF02C	£27.45
K-razy Shootout	-1E-8K-BQ63T	£34.95
*Moon Shuttle	-1C-16K-KF22Y	£27.50
Pac-Man	-1E-8K-BQ71N	£29.95
Paint	-1D-48K-KB22Y	£29.95
Picnic Paranoia	-1E-16K-KF13P	£34.95
*Preppie	-1C-16K-KB07H	£21.95
Qix	-1E-16K-KF16S	£29.95
SAM Speech Synth	-1D-32K-KB15R	£47.19
*Shamus	-1E-16K-KB90X	£34.95
Softporn Adventure	-1D-40K-BQ93B	£20.64
Starcross	-1D-32K-KB37S	£29.95
Synassembler	-1D-48K-KB83E	£38.95
*Zaxxon	-1C-16K-KF20W	£31.50
Zork I	-1D-32K-BQ94C	£29.95
Zork II	-1D-32K-BQ95D	£29.95
Zork III	-1D-32K-KB31J	£29.95
*3D-Supergraphics	-1C-40K-BQ29G	£31.95

Plus over 280 other titles for Atari.

COMMODORE 64

Temple Of Apshai (Part 1)	-1D-BC57M	£27.45
Upper Reaches (Part 2)	-1D-BC58N	£13.80
Curse Of Ra (Part 3)	-1D-BC59P	£13.80
Sword Of Fargoal	-1D-BC60Q	£20.75
Crush, Crumble & Chomp	-1D-BC61R	£20.75
Jump Man	-1D-BC62S	£27.45

DRAGON

Berserk	-1E-BC32K	£19.95
Black Sanctum	-1C-BC78K	£7.95
Dragon Trek	-1C-BC82D	£9.95
Galax Attax	-1E-BC79L	£19.95
Quest	-1C-BC41U	£7.95
Wizard War	-1C-BC83E	£7.95

Plus 30 other titles for Dragon

SPECTRUM

The Hobbit (48K)	-1C-BC88V	£14.95
Timegate (48K)	-1C-BC89W	£6.95
Space Intruders (16K)	-1C-BC90X	£4.95
Meteor Storm (16K)	-1C-BC91Y	£4.95
Chess Player (48K)	-1C-BC92A	£6.95
Speakeasy (48K)	-1C-BC93B	£4.95

VIC20

Crush, Crumble & Chomp (+16K)		
	1C-KK10L	£20.75
Datestones Of Ryn (+16K)	-1C-KK13P	£13.80
Invasion Orion (+16K)	-1C-KK12N	£17.25
Monster Maze	-1E-KK11M	£27.45
Plattermania	-1E-KK14Q	£27.45
Princess & Frog	-1E-KK16S	£29.95
Rescue At Rigel (+16K)	-1C-KK08J	£20.75
Ricochet (+8K)	-1C-KK15R	£13.80
Sword Of Fargoal (+16K)	-1C-KK09K	£20.75
Tank Arcade	-1C-KH18U	£11.95

Plus 80 other titles for VIC20

Disk versions also available though price and memory size may be different.

Send sae now for our new software leaflet with details of all programs added since Maplin catalogue.
Order As XH52G Issue 4.

MICROWRITER

The new hand-held word processor that eliminates the need for a typist. You can learn the Microwriting technique in less than an hour and produce perfectly typed text the same day.

Microwriter (AF62S) £485 + £72.75 VAT ★
Complete Word Processor Package (Microwriter, printer and lead) (AF63T) £1,455 + £218.25 VAT ★
Available ONLY by mail order or from WESTCLIFF shop.

★ Delivery next day by Datapost

ATARI®
AUTHORISED ATARI SERVICE CENTRE

INTEREST FREE CREDIT NOW!

If your order contains over £120 worth of computer hardware apply now for interest free credit by telephoning: Mail-order: (0702) 552911. London Shop: 01-248 0926. Birmingham Shop: 021-356 7292. Southend shop: 0702 554000 or write to P.O. Box 3, Rayleigh, Essex SS6 8LR.
You pay 10% down, then 10% per month for a further nine months (to nearest penny).

Credit quotations on request. This offer subject to approval which can take up to 48 hours (APR = 0%).

SORD M5
The brilliant new colour computer

A superb new home computer with one of the most powerful colour video processors available. An incredible 32 sprites simultaneously and 16K of RAM dedicated to graphics alone. Powerful Z80A (3.5MHz) main processor. Three tone generators and noise generator all with envelope control like a mini-synthesiser. Complete with BASIC and superb handbooks for beginners. All this for just £189.95 (AF64U)

Mapsoft full colour catalogue. Price £1 incl. post. Maplin catalogue contains full details of all hardware and lots of software. On sale now in all branches of W. H. Smith, price £1.25 or £1.50 incl. post from PO Box 3, Rayleigh, Essex.

Maplin Electronic Supplies Ltd., Mail Order: P.O. Box 3, Rayleigh, Essex SS6 8LR. Tel. Southend (0702) 552911 (Sales).
Demonstrations at our shops NOW.
159-161 King St., Hammersmith, W6. Tel. 01-748 0926. 284, London Road, Westcliff-on-Sea, Essex. Tel. (0702) 554000. Lynton Square, Perry Barr, Birmingham. Tel. (021) 356 7292.
All goods delivered in UK mainland carriage paid, but add 50p if total order less than £5 except catalogues. Orders including items marked ★, delivery next day by Datapost.
Subject to availability. All prices include VAT unless shown. Prices correct at time of going to press.

MONITOR AND DISASSEMBLER

These versatile routines written by Brian Cadge for the Dragon 32 provide you with a monitor which is ready for instant use whilst being invisible to the Basic user.

THIS MACHINE-CODE monitor for the Dragon 32 computer offers nine commands directly accessible from the keyboard in Basic. Once installed, the monitor is called by pressing the right arrow key; this is achieved by interrupting the Dragon's key-scan routine to check whether the right arrow has been pressed. In this way the monitor can be ready for use at any time whilst being transparent to the Basic user.

When you call the monitor by pressing the right arrow key, you should get the monitor's command prompt. This is a left-pointing arrow and means that Monit32 is expecting a command. There are two other important prompts which you will need to know, these are: the "greater-than sign" which means Monit32 requires a 16-bit — four digit hexadecimal — address or length. All numbers input and output in hexadecimal. It is important to enter four characters, for example if the number 7F3 were to be entered you must type 07F3.

The other prompt is a "less-than sign" which means that Monit32 is expecting an eight-bit — two-digit hex — number. Here is a summary of the commands and their uses:

O   Sets the origin address which is used by some of the other commands.

A   Allows you to enter code into memory starting at the address set by O. To terminate the command enter an asterisk as the code.

E   Displays the contents of a page of memory in hexadecimal starting from the address set by O. Pressing Enter will increase the address by one, pressing Clear will decrease the address by one. Pressing the space-bar returns to command level.

F   This command is used to fill a section of memory with a constant. When pressed, Monit32 will first require a 16-bit length and then the value of the constant. That length of memory will then be filled with that constant starting at the address set by O.

J   This is used to jump to the 16-bit address given after J.

M   The M command is very useful for moving machine-code or graphics pages. When called, the monitor first requires the length of code to be moved, for example, 00FE to move 254 bytes. Then the starting address of the destination is entered. The code starting at the address set by O will then be copied to the memory starting at the address given for the length given.

R   The contents of the 6809E microprocessor registers, A, B, X, Y, S and U

Disassembled version of the monitor.
```
10 CLEAR 200,31699
15 T=0:PRINT:PRINT"PLEASE WAIT A MOMENT WHILST CODEIS INSTALLED..." 'SOUND190,1
20 FOR I=1 TO 59:READ A$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
21 IF T<>5885 THEN PRINT"DATA ERROR IN 30":STOP
30 DATA 81,09,27,1,39,34,36,86,39,B7,01,6A,9E,88,86,5F,A7,89,0,0,30,1,9F,88,BD,A
0,EA,B7,7F,EE,8A,40,9E,88,A7,89,0,0,30,1,9F,88,B6,7F,EE,81,5A,26,08,86,7E,B7,01,
6A,35,36,39,20,3B
50 'return value in buff+X in B
60 FOR I=60 TO 118:READ A$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
61 IF T<>13250 THEN PRINT"DATA ERROR IN 70":STOP
70 DATA A6,89,7F,EE,80,39,2E,0B,A6,89,7F,EE,80,30,C6,10,3D,20,08,A6,89,7F,EE,80,
37,20,F3,A6,89,7F,EF,80,39,2E,0F,A6,89,7F,EF,80,30,12,B7,7F,F2,FB,7F,F2,20,08,A6
,89,7F,EF,80,37,20,EF,39
90 'input routine for buff1-4
100 FOR I=119 TO 183:READA$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
101 IF T<>19267 THEN PRINT"DATA ERROR IN 110":STOP
110 DATA 7E,7D,E9,8E,0,0,BD,A0,EA,81,08,26,11,8C,0,0,27,F4,10,9E,88,31,3F,10,9F,
88,30,1F,20,E8,81,0D,26,04,BD,90,A1,39,A7,89,7F,EE,8A,40,8C,00,04,24,0E,30,01,10
,9E,88,A7,A9,00,00,31,21,10,9F,88,20,C5
150 'hex print routine val in B
160 FOR I=184 TO 233:READ A$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
161 IF T<>24421 THEN PRINT"DATA ERROR IN 170":STOP
170 DATA F7,7F,FF,54,54,54,54,C4,0F,C1,09,2E,04,CB,70,20,02,CB,37,8D,12,F6,7F,FF
,C4,0F,C1,09,2E,04,CB,70,20,02,CB,37,8D,01,39,9E,88,E7,89,00,00,30,01,9F,88,39
190 'recognise MONIT32 command
200 FOR I=234 TO 307:READ A$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
201 IF T<>31181 THEN PRINT"DATA ERROR IN 210":STOP
210 DATA 81,4F,26,03,7E,7E,33,81,4A,26,03,7E,7E,57,81,53,26,03,7E,7E,8A,81,45,26
,3,7E,7E,95,81,46,26,03,7E,7F,1E,81,4D,26,03,7E,7F,66,81,52,26,03,7E,7C,3B,81,41
,26,03,7E,7B,D4,9E,88,86,7F,A7,89,0,0,30,1,9F,88,BD,90,A1,7E,7D,0C
250 'set origin module 'O'
260 FOR I=308 TO 343:READ A$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
261 IF T<>35368 THEN PRINT"DATA ERROR IN 270":STOP
270 DATA 9E,88,86,7E,A7,89,0,0,30,01,9F,88,BD,7D,79,8E,0,0,BD,7D,3B,F7,7F,FD,8E,
0,2,BD,7D,3B,F7,7E,FE,7E,7D,0C
290 'Jump to address 'J'
300 FOR I=344 TO 394:READA$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
301 IF T<>40535 THEN PRINT"DATA ERROR IN 310":STOP
310 DATA 9E,88,86,7E,A7,89,0,0,30,1,9F,88,BD,7D,79,8C,0,0,2E,3,7E,7D,0C,8E,0,0,B
D,7D,3B,F7,7E,88,8E,0,2,BD,7D,3B,F7,7E,89,86,7E,B7,01,6A,35,36,7E,83,71
```

```
340 'exec subroutine from origin
350 FORI=395 TO 405:READA$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
351 IF T<>42167 THEN PRINT"DATA ERROR IN 360":STOP
360 DATA BD,90,A1,BE,7F,FD,AD,84,7E,7D,0C
390 'examine memory form origin
400 FOR I=406 TO 542:READ A$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
401 IF T<>59267 THEN PRINT"DATA ERROR IN 410/420":STOP
410 DATA 8E,0,0,86,60,A7,89,04,00,30,01,8C,02,00,26,F5,8E,04,00,9F,88,10,BE,7F
,D,10,BF,7F,F6,86,0E,B7,7F,F3,F6,7F,F6,BD,7D,B7,F6,7F,F7,BD,7D,B7,86,09,9E,88,30,
01,9F,88,10,BE,7F,F6,E6,A4,31,21,10,BF,7F,F6,BD,7D,B7,4A,26,E8,BD,90,A1,B6,7F,F5
,4A
420 DATA 27,05,B7,7F,F5,20,CC,9E,88,86,6A,A7,84,30,1,9F,88,BD,A0,EA,81,0D,26,0A,
BE,7F,FD,30,01,BF,7F,FD,20,F0,81,0C,26,0A,BE,7F,FD,30,1F,BF,7F,FD,20,F0,81,20,26
,DD,BD,90,A1,7E,7D,0C
450 'fill form origin 'F'
460 FOR I=543 TO 614:READ A$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
461 IF T<>67688 THEN PRINT"DATA ERROR IN 470":STOP
470 DATA 86,7E,9E,88,A7,84,30,01,9F,88,BD,7D,79,8E,0,0,BD,7D,3B,F7,7F,F5,8E,00,0
2,BD,7D,3B,F7,7F,F6,9E,88,86,7C,A7,84,30,01,9F,88,BD,7D,79,8E,0,0,BD,7D,3B,BE,7F
,FD,10,BE,7F,F5,F6,84,30,01,31,3F,10,8C,0,0,26,F4,7E,7D,0C
490 'move code 'M'
500 FOR I=615 TO 714:READ A$:POKE 31999+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
501 IF T<>79939 THEN PRINT"DATA ERROR IN 510/520":STOP
510 DATA 86,7E,9E,88,A7,84,30,01,9F,88,BD,7D,79,8E,0,0,BD,7D,3B,F7,7F,F5,8E,0,2,
BD,7D,3B,F7,7F,F6,9E,88,86,7E,9E,88,A7,84,30,01,9F,88,BD,7D,79,8E,0,0,BD,7D,3B,F
7,7F,8E,0,2,BD,7D,3B,F7,7F,F8,BE,7F,FD,10,BE,7F,F5,A6,84,A7,A4,30,01,31,3F,10,BF
,F7
520 DATA 10,BE,7F,F5,31,3F,10,BF,7F,F5,10,8C,0,0,26,E0,7E,7D,0C
550 'display registers 'R'
560 FOR I=1TO197:READA$:POKE 31802+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
561 IF T<>108885 THEN PRINT"DATA ERROR IN 570/580/590":STOP
570 DATA 35,36,BF,7F,EE,10,BF,7F,F0,10,FF,7F,F2,FF,7F,F4,B7,7F,F6,F7,7F,F7,34,36
,BD,90,A1,20,16,34,2,BD,90,A1,35,2,9E,88,A7,84,30,A7,84,30,02,9F,88,39,
86,41,8D,E6,F6,7F,F6,BD,7D,B7,86,42,8D,DC,F6,7F,F7,BD,7D,B7,86,58,8D,D2,F6,7F,EE
,BD,7D,B7
580 DATA F6,7F,EF,BD,7D,B7,86,59,8D,C2,F6,7F,F0,BD,7D,B7,F6,7F,F1,BD,7D,B7,86,53
,8D,B2,F6,7F,F2,BD,7D,B7,F6,7F,F3,BD,7D,B7,86,55,8D,A2,F6,7F,F4,BD,7D,B7,F6,7F,F
5,BD,7D,B7,BD,90,A1,BD,90,A1,9E,88,86,50,A7,84,30,1,86,52,A7,84,30,1,86,4F,A7,84
,30,1,86
590 DATA 47,A7,84,30,2,9F,88,DC,1B,83,0,2,93,19,FD,7F,EE,F6,7F,EE,BD,7D,B7,F6,7F
,EF,BD,7D,B7,BD,90,A1,7E,7D,0C
600 'enter code into memory 'E'
610 FOR I=1 TO 103:READA$:POKE 31699+I,VAL("&H"+A$):T=T+VAL("&H"+A$):NEXT
611 IF T<>122108 THEN PRINT"DATA ERROR IN 620/630"
620 DATA BD,90,A1,9E,88,86,4F,A7,84,30,01,86,7D,A7,84,30,02,9F,88,F6,7F,FD,BD,7D
,B7,F6,7F,FE,BD,7D,B7,86,A1,10,BE,7F,FD,10,BF,7F,F6,9E,88,86,7C,A7,84,30,02,9
F,88,BD,7D,79,B6,7F,EE,81,2A,26,03,7E,7D,0C,8C,0,0,26,0F,9E,88,86,7F,A7,84,30,1,
9F,88
630 DATA BD,90,A1,20,D5,8E,0,0,BD,7D,3B,BE,7F,F6,E7,84,30,1,BF,7F,F6,20,C3
700 POKE 363,125:POKE 364,0
710 POKE 362,126
720 CLS:PRINT"MONIT32 INSTALLED AND READY -   PRESS RIGHT ARROW KEY TO CALL   MO
NIT32...":SOUND190,1
730 NEW
```

Program 2.

```
5 ' RE-PROGRAM RESET BUTTON
10 CLEAR 300,31099
20 FOR I=0 TO 22:POKE 31100+I,PEEK(46159+I):NEXT
30 FOR I=23 TO 48:READA$:POKE 31100+I,VAL("&H"+A$):NEXT
40 DATA 8E,0,0,C6,01,A6,89,79,AE,A7,89,04,00,30,01,5A,26,F3,8E,04,FF,9F,88,7E,83
,71
50 CLS
60 PRINT "ENTER MESSAGE TO BE PRINTED WHENRESET BUTTON IS PRESSED (NO    INVERS
E CHARACTERS...":PRINT
70 LINE INPUT A$
80 FOR I=1 TO LEN(A$)
90 POKE 31149+I,ASC(MID$(A$,I,1))OR64
100 NEXT:POKE 31127,LEN(A$)
110 POKE 114,121:POKE 115,124
120 PRINT:PRINT"MESSAGE ENTERED... PRESS RESET   TO TEST..."
130 END
```

Program 3.

```
10 ' 6809 DIS-ASSEMBLER FOR DRAGON 32 BY BRIAN CADGE
20 ' ************************************
30 CLS:POKE 155,80:POKE 154,64:POKE 153,16:POKE 328,0  'SET UP 80 COLUMN PRINTER
40 CLEAR 5000:DIM A$(255,2)
50 FOR I=0 TO 255:READ A$(I,0),A$(I,1),A$(I,2):NEXTI
60 PRINT"6809 DIS-ASSEMBLER FOR DRAGON 32"
70 PRINT:PRINT:PRINT:SOUND100,1:SOUND150,1:SOUND200,1
80 PS=PEEK(65314)AND1
90 INPUT "PRINTER [P] OR SCREEN [S] ";OP$:IF OP$<>"P" AND OP$<>"S" THEN 90 ELSE
IF OP$="P" THEN PR=-2 ELSE PR=0
100 IF PR=-2 AND PS=1 THEN PRINT"ATTEND TO PRINTER!!!":EXEC 41194:GOTO 80
110 INPUT "START ADDRESS , END ADDRESS";AD,EA
120 IF EA<0 OR EA>65535 OR EA<AD OR AD<0 OR AD>65535 THEN PRINT"INVALID ADDRESS
- REDO":GOTO70
130 CLS:PRINT@500,:PRINT#PR,"":PRINT#PR,""
140 FOR I=AD TO EA
150 LI=I
160 SH=0
170 V=PEEK(I)
180 IF V=16 THEN SH=1:I=I+1:GOTO170
190 IF V=17 THEN SH=2:I=I+1:GOTO170
200 Z$=A$(V,SH):X$=RIGHT$(Z$,1)
210 IF X$<>"^" AND X$<>"$" AND X$<>"%" AND X$<>"&" AND X$<>"<" AND X
$<>">" AND X$<>"/" THEN P$=Z$ ELSE P$=LEFT$(Z$,LEN(Z$)-1)
220 IF PR=0 THEN PRINTI;TAB(10);P$; ELSE PRINT#PR,I,P$;
230 IF X$="#" THEN 360
240 IF X$="%" THEN 380
```

will be displayed, together with the length of any Basic program currently in the memory.

S This command executes a machine-code subroutine starting at the address given by O and returns to the monitor.

Z Returns to Basic.

The program is used to load to monitor in reserved memory at the top of RAM. When run it will check for errors in the Data statements and inform you of any, in which case you should check through your listing against the one given. If there are no errors the monitor is installed and the loading program is Newed.

As previously mentioned, the program interrupts the key-scan routine to check if the arrow key has been pressed. Anybody interested in writing programs to use this — for example, a keyboard beep routine — will need to know the following.

Locations 363 and 364 contain the address to which the Dragon jumps when a key is pressed. To enable the interrupt, location 362 is Poked with 126, to disable the interrupt a value of 57 is Poked to location 362.

The second short program demonstrates how the Reset button can also be interrupted when pressed. The program asks for a message to be typed in and this message will then be printed before the OK prompt when ever the Reset key is pressed. This is achieved by changing locations 114 and 115 which point to the address to which the Dragon will jump when the Reset key is pressed.

It is important that the machine recovers correctly, so part of the Reset routine is copied from the ROM first of all in line 20, then your own machine-code program can follow this.

Finally, one very useful command that is missing from the Dragon's Basic is the Merge facility. However, this can be achieved as follows: type Cload "programone". When loaded type

? PEEK(25);PEEK(26);PEEK(27);PEEK(28)

Make a note of the four numbers displayed. Type

POKE 25, PEEK(27) : POKE 26, PEEK(28)-2

Now Cload "programtwo". Renumber it so that its first line number is higher than the last line number of the first program. Finally, Poke locations 25 and 26 with the original values. The programs should now have been merged. This works as locations 25 and 26 contain the Start address of the program and locations 27 and 28 contain the end address plus two of the program. This information can also be used to clear all of the graphics pages; usually you must have at least one page reserved. Type

POKE 25,6 : NEW

and you will have just under 31K of free memory. Make sure that there is no program in the memory before you do this or it will create chaos.

Program 3 is a full 6809 disassembler program designed to run on the Dragon 32. Note that the program can be used with or without the printer, the only difference in output being that the hexadecimal code is also output to the printer whereas only the address and mnemonic is printed when disassembling to the screen.

All of the mnemonics are held in Data

statements at the end of the program in lines 890 onwards. Take care when entering these to insert the correct number of commas between each word as this is very important.

The symbol after each mnemonic signifies to which addressing mode, if any, the op-code refers; for example a "greater than" symbol means indexed addressing.

The main loop is from lines 140 to 350, this prints the correct mnemonic and transfers control to one of the subroutines to print any following data.

The use of Print with a channel number means that exactly the same program can be used for both the screen and printer. Line 90 sets PR to 0 for the screen, or −2 for the printer.

Line 30 sets up an 80-column printer, and although Dragon Data never mention this, this line should be included with any program that is going to use a printer.

Use Shift @ to halt the disassembly as the text scrolls up the screen at quite a pace. If you use the program to disassemble the Basic ROM it is quite easy to find some very useful subroutines that can be called from your own Basic or machine-code programs. For example, try entering the starting address as 48299 and you will get a disassembly of the ROM's print routine, which prints the character in register A.

```
250 IF X$="$" THEN 400
260 IF X$="&" THEN 430
270 IF X$=">" THEN 460
280 IF X$="<" THEN 680
290 IF X$="^" THEN 750
300 IF X$="/" THEN 820
310 IF PR=0 THEN 350
320 PRINT#PR,TAB(36)::FOR JJ=LI TO I:HE$=HEX$(PEEK(JJ)):IF LEN(HE$)<2 THEN HE$="0"+HE$
330 PRINT#PR,HE$;
340 NEXT JJ
350 PRINT#PR,"":NEXTI:END
360 I=I+1:V=PEEK(I):PRINT#PR,V;
370 GOTO 310
380 I=I+1:V=PEEK(I)*256+PEEK(I+1):I=I+1:PRINT#PR,V;
390 GOTO 310
400 I=I+1:V=PEEK(I)
410 PRINT#PR,"("V")";
420 GOTO 310
430 I=I+1:V=PEEK(I)*256+PEEK(I+1):I=I+1
440 PRINT#PR,"("V")";
450 GOTO310
460 I=I+1
470 V=PEEK(I)
480 PRINT#PR,"[";
490 V=V AND 96
500 IF V=0 THEN P$="X"
510 IF V=32 THEN P$="Y"
520 IF V=64 THEN P$="U"
530 IF V=96 THEN P$="S"
540 PRINT#PR,P$;" ";
550 IF (PEEK(I)AND128)=0 THEN PRINT#PR,"+";(PEEK(I)AND31);"]";:GOTO670
560 V=PEEK(I)AND159
570 IF V=132 THEN PRINT#PR," ]":GOTO670
580 IF V=136 THEN PRINT#PR,"+";:GOTO360
590 IF V=137 THEN PRINT#PR,"+";:GOTO380
600 IF V=134 THEN PRINT#PR,"+A]":GOTO670
610 IF V=133 THEN PRINT#PR,"+B]":GOTO670
620 IF V=139 THEN PRINT#PR,"+D]":GOTO670
630 IF V=128 THEN PRINT#PR," INC 1]";
640 IF V=129 THEN PRINT#PR," INC 2]";
650 IF V=130 THEN PRINT#PR," DEC 1]";
660 IF V=131 THEN PRINT#PR," DEC 2]";
670 GOTO310
680 I=I+1:V=PEEK(I)
690 P$=""
700 FOR J=7 TO 0 STEP-1
710 IF V<INT(2^J) THEN P$=P$+"0" ELSE V=V-INT(2^J):P$=P$+"1"
720 NEXTJ
730 PRINT#PR,P$" BIN";
740 GOTO310
750 I=I+1:V=PEEK(I)
760 IF V>127 THEN 790
770 PRINT#PR,"+"V;
780 GOTO810
790 V=256-V
800 PRINT#PR,"-"V;
810 GOTO 310
820 I=I+1:V=PEEK(I)*256+PEEK(I+1):I=I+1
830 IF V>32767 THEN 860
840 PRINT#PR,"+"V;
850 GOTO880
860 V=65536-V
870 PRINT#PR,"-"V;
880 GOTO310
890 DATA NEG $,,,ERR,,,ERR,,,COM $,,,LSR $,,,ERR,,,ROR $,,,ASR $,,,ASL $,,,ROL $,,,DEC $,,,ERR,,,INC $,,,TST $,,,JMP $,,,CLR $,,,SFT,,,SFT2,,,NOP,,,SYNC,,,ERR,,,ERR,,,ERR,,,LBRS %,,
900 DATA ERR,,,DAA,,,OR CC #,,,ERR,,,AND CC #,,,SEX,,,EXG <,,,TFR <,,,BRA ^,LBRA /,,BRN ^,LBRN /,,BHI ^,LBHI /,,BLS ^,LBLS /,,BHS ^,LBHS /,,BLO ^,LBLO /,,BNE ^,LBNE /,,BEQ ^,LBEQ /,,BVC ^,LBVC /,,BVS ^,LBVS /,,BPL ^,LBPL /,,BMI ^,LBMI /,,BGE ^,LBGE /,,
910 DATA BLT ^,LBLT /,,BGT ^,LBGT /,,BLE ^,LBLE /,,LEA X >,,,LEA Y >,,,LEA S >,,,LEA U >,,,PSHS <,,,PULS <,,,PSHU <,,,PULU <,,,ERR,,,RTS,,,ERR,,,RTI,,,CWAI #,,,MUL,,,ERR,,,SWI 1,SWI 2,SWI 3,NEG A,,,ERR,,,ERR,,,COM A,,
920 DATA LSR A,,,ERR,,,ROR A,,,ASR A,,,ASL A,,,ROL A,,,DEC A,,,ERR,,,INC A,,,TST A,,,ERR,,,CLR A,,,NEG B,,,ERR,,,ERR,,,COM B,,,LSR B,,,ERR,,,ROR B,,,ASR B,,,ASL B,,,ROL B,,,DEC B,,,ERR,,,INC B,,,TST B,,,ERR,,,CLR B,,,NEG >,,,ERR,,,ERR,,,COM >,,
930 DATA LSR >,,,ERR,,,ROR >,,,ASR >,,,ASL >,,,ROL >,,,DEC >,,,ERR,,,INC >,,,TST >,,,JMP >,,,CLR >,,,NEG &,,,ERR,,,ERR,,,COM &,,,LSR &,,,ERR,,,ROR &,,,ASR &,,,ASL &,,,ROL &,,,DEC &,,,ERR,,,INC &,,,TST &,,,JMP &,,,CLR &,,,SUB A #,,,CMP A #,,,SBC A #,,
940 DATA SUB D %,CMP D %,CMP U %,AND A #,,,BIT A #,,,LDA #,,,ERR,,,EOR A #,,,ADC A #,,,OR A #,,,ADD A #,,,CMP X %,CMP Y %,CMP S %,BSR ^,,,LDX %,LDY %,,,ERR,,,SUB A $,,,CMP A $,,,SBC A $,,,SUB D $,CMP D $,CMP U $,AND A $,,,BIT A $,,,LDA $,,,STA $,,
950 DATA EOR A $,,,ADC A $,,,OR A $,,,ADD A $,,,CMP X $,CMP Y $,CMP S $,JSR $,,,LDX $,LDY $,,STX $,STY $,,SUB A >,,,CMP A >,,,SBC A >,,,SUB D >,CMP D >,CMP U >,AND A >,,,BIT A >,,,LDA >,,,STA >,,,EOR A >,,,ADC A >,,,OR A >,,,ADD A >,,
960 DATA CMP X >,CMP Y >,CMP S >,JSR >,,,LDX >,LDY >,,STX >,STY >,,SUB A &,,,CMP A &,,,SBC A &,,,SUB D &,CMP D &,CMP U &,AND A &,,,BIT A &,,,LDA &,,,STA &,,,EOR A &,,,ADC A &,,,OR A &,,,ADD A &,,,CMP X &,CMP Y &,CMP S &,JSR &,,,LDX &,LDY &,,
970 DATA STX &,STY &,,SUB B #,,,CMP B #,,,SBC B #,,,ADD D %,AND B #,,,BIT B #,,,LDB #,,,ERR,,,EOR B #,,,ADC B #,,,OR B #,,,ADD B #,,,LDD %,,ERR,,,LDU %,LDS %,,ERR,,,CMP B $,,,SBC B $,,,ADD D $,,,AND B $,,,BIT B $,,
980 DATA LDB $,,,STB $,,,EOR B $,,,ADC B $,,,OR B $,,,ADD B $,,,LDD $,,,STD $,,,LDU $,LDS $,,STU $,STS $,,SUB B >,,,CMP B >,,,SBC B >,,,ADD D >,,,AND B >,,,BIT B >,,,LDB >,,,STB >,,,EOR B >,,,ADC B >,,,OR B >,,,ADD >,,,LDD >,,,STD >,,,LDU >,,LDS >,
990 DATA STU >,STS >,,SUB B &,,,CMP B &,,,SBC B &,,,ADD A &,,,AND B &,,,BIT B &,,,LDB &,,,STB &,,,EOR B &,,,ADC B &,,,OR B &,,,ADD B &,,,LDD &,,,STD &,,,LDU &,LDS &,,STU &,STS &,
```

```
31700        JSR ( 37025 )BD90A1
31703        LDX ( 136 )     9E88
31705        LDA 79          864F
31707        STA [X]         A784
31709        LEA X 1         3001
31711        LDA 125         867D
31713        STA [X]         A784
31715        LEA X 2         3002
31717        STX ( 136 )     9F88
31719        LDB ( 32765 )   F67FFD
31722        JSR ( 32183 )   BD7DB7
31725        LDB ( 32766 )   F67FFE
31728        JSR ( 32183 )   BD7DB7
31731        JSR ( 37025 )   BD90A1
31735        LDY ( 32765 )   10BE7FFD
31739        STY ( 32758 )   10BF7FF6
31742        LDX ( 136 )     9E88
31744        LDA 124         867C
31746        STA [X]         A784
31748        LEA X 2         3002
31750        STX ( 136 )     9F88
31752        JSR ( 32121 )   BD7D79
31755        LDA ( 32750 )   B67FEE
31758        CMP A 42        812A
31760        BNE + 3         2603
31762        JMP ( 32012 )   7E7D0C
31765        CMP X 0         8C0000
31768        BNE + 15        260F
31770        LDX ( 136 )     9E88
31772        LDA 127         867F
31774        STA [X]         A784
31776        LEA X 1         3001
31778        STX ( 136 )     9F88
31780        JSR ( 37025 )   BD90A1
31783        BRA - 43        20D5
31785        LDX 0           8E0000
31788        JSR ( 32059 )   BD7DBB
31791        LDX ( 32758 )   BE7FF6
31794        STB [X]         E784
31796        LEA X 1         3001
31798        STX ( 32758 )   BF7FF6
31801        BRA - 61        20C3
31803        PULS 00110110 BIN 3536
31805        STX ( 32750 )   BF7FEE
31809        STY ( 32752 )   10BF7FF0
31813        STS ( 32754 )   10FF7FF2
31816        STU ( 32756 )   FF7FF4
31819        STA ( 32758 )   B77FF6
31822        STB ( 32759 )   F77FF7
31825        PSHS 00110110 BIN 3435
31827        JSR ( 37025 )   BD90A1
31830        BRA + 22        2016
31832        PSHS 00000010 BIN 3402
31834        JSR ( 37025 )   BD90A1
31837        PULS 00000010 BIN 3502
31839        LDX ( 136 )     9E88
31841        STA [X]         A784
31843        LEA X 1         3001
31845        LDA 125         867D
31847        STA [X]         A784
31849        LEA X 2         3002
31851        STX ( 136 )     9F88
31853        RTS             39
31854        LDA 65          8641
31856        BSR - 26        8DE6
```
(continued on next page)

```
31858  LDB ( 32758 )      F67FF6
31861  JSR ( 32183 )      BD7DB7
31864  LDA 66             8656
31866  BSR - 36           8DDC
31868  LDB ( 32759 )      F67FF7
31871  JSR ( 32183 )      BD7DB7
31874  LDA 88             865B
31876  BSR - 46           8DD2
31878  LDB ( 32750 )      F67FEE
31881  JSR ( 32183 )      BD7DB7
31884  LDB ( 32751 )      F67FEF
31887  JSR ( 32183 )      BD7DB7
31890  LDA 89             8659
31892  BSR - 62           8DC2
31894  LDB ( 32752 )      F67FF0
31897  JSR ( 32183 )      BD7DB7
31900  LDB ( 32753 )      F67FF1
31903  JSR ( 32183 )      BD7DB7
31906  LDA 83             8653
31908  BSR - 78           8DB2
31910  LDB ( 32754 )      F67FF2
31913  JSR ( 32183 )      BD7DB7
31916  LDB ( 32755 )      F67FF3
31919  JSR ( 32183 )      BD7DB7
31922  LDA 85             8655
31924  BSR - 94           8DA2
31926  LDB ( 32756 )      F67FF4
31929  JSR ( 32183 )      BD7DB7
31932  LDB ( 32757 )      F67FF5
31935  JSR ( 32183 )      BD7DB7
31938  JSR ( 37025 )      BD90A1
31941  JSR ( 37025 )      BD90A1
31944  LDX ( 136 )        9E88
31946  LDA 80             8650
31948  STA [x]            A784
31950  LEA X 1            3001
31952  LDA 82             8652
31954  STA [x]            A784
31956  LEA X 1            3001
31958  LDA 79             864F
31960  STA [x]            A784
31962  LEA X 1            3001
31964  LDA 71             8647
31966  STA [x]            A784
31968  LEA X 2            3002
31970  STX ( 136 )        9F88
31972  LDD ( 27 )         DC1B
31974  SUB D 2            830002
31977  SUB D ( 25 )       9319
31979  STD ( 32750 )      FD7FEE
31982  LDB ( 32750 )      F67FEE
31985  JSR ( 32183 )      BD7DB7
31988  LDB ( 32751 )      F67FEF
31991  JSR ( 32183 )      BD7DB7
31994  JSR ( 37025 )      BD90A1
31997  JMP ( 32012 )      7E7D0C
32000  CMP A 9            8109
32002  BEQ + 1            2701
32004  RTS                39
32005  PSHS 00110110 BIN  3436
32007  LDA 57             8639
32009  STA ( 362 )        B7016A
32012  LDX ( 136 )        9E88
32014  LDA 95             865F
32016  STA [x + 0 ]       A7890000
32020  LEA X 1            3001
32022  STX ( 136 )        9F88
32024  JSR ( 41194 )      BDA0EA
32027  STA ( 32750 )      B77FEE
32030  OR A 64            8A40
32032  LDX ( 136 )        9E88
32034  STA [x + 0 ]       A7890000
32038  LEA X 1            3001
32040  STX ( 136 )        9F88
32042  LDA ( 32750 )      B67FEE
32045  CMP A 90           815A
32047  BNE + 8            2608
32049  LDA 126            867E
32051  STA ( 362 )        B7016A
32054  PULS 00110110 BIN  3536
32056  RTS                39
32057  BRA + 59           203B
32059  LDA [x + 32750 ]   A6897FEE
32063  SUB A 57           8039
32065  BGT + 11           2E08
32067  LDA [x + 32750 ]   A6897FEE
32071  SUB A 48           8030
32073  LDB 16             C610
32075  MUL                3D
32076  BRA + 8            2008
32078  LDA [x + 32750 ]   A6897FEE
32082  SUB A 55           8037
32084  BRA - 13           20F3
32086  LDA [x + 32751 ]   A6897FEF
32090  SUB A 57           8039
32092  BGT + 15           2E0F
32094  LDA [x + 32751 ]   A6897FEF
32098  SUB A 48           8030
32100  NOP                12
32101  STA ( 32754 )      B77FF2
32104  ADD B ( 32754 )    FB7FF2
32107  BRA + 8            2008
32109  LDA [x + 32751 ]   A6897FEF
32113  SUB A 55           8037
32115  BRA - 17           20EF
32117  RTS                39
32118  JMP ( 32233 )      7E7DE9
32121  LDX 0              8E0000
32124  JSR ( 41194 )      BDA0EA
32127  CMP A 8            8108
32129  BNE + 17           2611
32131  CMP X 0            8C0000
32134  BEQ - 12           27F4
32137  LDY ( 136 )        109E88

32139  LEA Y 63           313F
32142  STY ( 136 )        109F88
32144  LEA X 31           301F
32146  BRA - 24           20E8
32148  CMP A 13           810D
32150  BNE + 4            2604
32152  JSR ( 37025 )      BD90A1
32155  RTS                39
32156  STA [x + 32750 ]   A7897FEE
32160  OR A 64            8A40
32162  CMP X 4            8C0004
32165  BHS + 14           240E
32167  LEA X 1            3001
32170  LDY ( 136 )        109E88
32172  STA [w + 0 ]       A7A90000
32176  LEA Y 33           3121
32179  STY ( 136 )        109F88
32181  BRA - 59           20C5
32183  STB ( 32767 )      F77FFF
32186  LSR B              54
32187  LSR B              54
32188  LSR B              54
32189  LSR B              54
32190  AND B 15           C40F
32192  CMP B 9            C109
32194  BGT + 4            2E04
32195  ADD B 112          CB70
32198  BRA + 2            2002
32200  ADD B 55           CB37
32202  BSR + 18           8D12
32204  LDB ( 32767 )      F67FFF
32207  AND B 15           C40F
32209  CMP B 9            C109
32211  BGT + 4            2E04
32213  ADD B 112          CB70
32215  BRA + 2            2002
32217  ADD B 55           CB37
32219  BSR + 1            8D01
32221  RTS                39
32222  LDX ( 136 )        9E88
32224  STB [x + 0 ]       E7890000
32228  LEA X 1            3001
32230  STX ( 136 )        9F88
32232  RTS                39
32233  CMP A 79           814F
32235  BNE + 3            2603
32237  JMP ( 32307 )      7E7E33
32240  CMP A 74           814A
32242  BNE + 3            2603
32244  JMP ( 32343 )      7E7E57
32247  CMP A 83           8153
32249  BNE + 3            2603
32251  JMP ( 32394 )      7E7E8A
32254  CMP A 69           8145
32256  BNE + 3            2603
32258  JMP ( 32405 )      7E7E95
32261  CMP A 70           8146
32263  BNE + 3            2603
32265  JMP ( 32542 )      7E7F1E
32268  CMP A 77           814D
32270  BNE + 3            2603
32272  JMP ( 32614 )      7E7F66
32275  CMP A 82           8152
32277  BNE + 3            2603
32279  JMP ( 31803 )      7E7C3B
32282  CMP A 65           8141
32284  BNE + 3            2603
32286  JMP ( 31700 )      7E7BD4
32289  LDX ( 136 )        9E88
32291  LDA 127            867F
32293  STA [x + 0 ]       A7890000
32297  LEA X 1            3001
32299  STX ( 136 )        9F88
32301  JSR ( 37025 )      BD90A1
32304  JMP ( 32012 )      7E7D0C
32307  LDX ( 136 )        9E88
32309  LDA 126            867E
32311  STA [x + 0 ]       A7890000
32315  LEA X 1            3001
32317  STX ( 136 ) —      9F88
32319  JSR ( 32121 )      BD7D79
32322  LDX 0              8E0000
32325  JSR ( 32059 )      BD7D3B
32328  STB ( 32765 )      F77FFD
32331  LDX 2              8E0002
32334  JSR ( 32059 )      BD7D3B
32337  STB ( 32766 )      F77FFE
32340  JMP ( 32012 )      7E7D0C
32343  LDX ( 136 )        9E88
32345  LDA 125            867E
32347  STA [x + 0 ]       A7890000
32351  LEA X 1            3001
32353  STX ( 136 )        9F88
32355  JSR ( 32121 )      BD7D79
32358  CMP X 0            8C0000
32361  BGT + 3            2E03
32363  JMP ( 32012 )      7E7D0C
32366  LDX 0              8E0000
32369  JSR ( 32059 )      BD7D3B
32372  STB ( 32392 )      F77E88
32375  LDX 2              8E0002
32378  JSR ( 32059 )      BD7D3B
32381  STB ( 32393 )      F77E89
32384  LDA 126            867E
32386  STA ( 362 )        B7016A
32389  PULS 00110110 BIN  3536
32391  JMP ( 33649 )      7E8371
32394  JSR ( 37025 )      BD90A1
32397  LDX ( 32765 )      BE7FFD
32400  JSR [x]            AD84
32402  JMP ( 32012 )      7E7D0C
32405  LDX 0              8E0000
32408  LDA 96             8660
32410  STA [x + 1024 ]    A7890400
32414  LEA X 1            3001

32416  CMP X 512          8C0200
32419  BNE - 11           26F5
32421  LDX 1024           8E0400
32424  STX ( 136 )        9F88
32427  LDY ( 32765 )      10BE7FFD
32431  STY ( 32758 )      10BF7FF6
32434  LDA 14             860E
32436  STA ( 32757 )      B77FF5
32439  LDB ( 32758 )      F67FF6
32442  JSR ( 32183 )      BD7DB7
32445  LDB ( 32759 )      F67FF7
32448  JSR ( 32183 )      BD7DB7
32451  LDA 9              8609
32453  LDX ( 136 )        9E88
32455  LEA X 1            3001
32457  STX ( 136 )        9F88
32460  LDY ( 32758 )      10BE7FF6
32464  LDB [x]            E6A4
32466  LEA Y 33           3121
32468  STY ( 32758 )      10BF7FF6
32471  JSR ( 32183 )      BD7DB7
32474  DEC A              4A
32475  BNE - 24           26E8
32477  JSR ( 37025 )      BD90A1
32480  LDA ( 32757 )      B67FF5
32483  DEC A              4A
32484  BEQ + 5            2705
32486  STA ( 32757 )      B77FF5
32489  BRA - 52           20CC
32491  LDX ( 136 )        9E88
32493  LDA 106            866A
32495  STA [x]            A784
32497  LEA X 1            3001
32499  STX ( 136 )        9F88
32501  JSR ( 41194 )      BDA0EA
32504  CMP A 13           810D
32506  BNE + 10           260A
32508  LDX ( 32765 )      BE7FFD
32511  LEA X 1            3001
32513  STX ( 32765 )      BF7FFD
32516  BRA - 97           209F
32518  CMP A 12           810C
32520  BNE + 10           260A
32522  LDX ( 32765 )      BE7FFD
32525  LEA X 31           301F
32527  STX ( 32765 )      BF7FFD
32530  BRA - 16           20F0
32532  CMP A 32           8120
32534  BNE - 35           26DD
32536  JSR ( 37025 )      BD90A1
32539  JMP ( 32012 )      7E7D0C
32542  LDA 126            867E
32544  LDX ( 136 )        9E88
32546  STA [x]            A784
32548  LEA X 1            3001
32550  STX ( 136 )        9F88
32552  JSR ( 32121 )      BD7D79
32555  LDX 0              8E0000
32558  JSR ( 32059 )      BD7D3B
32561  STB ( 32757 )      F77FF5
32564  LDX 2              8E0002
32567  JSR ( 32059 )      BD7D3B
32570  STB ( 32758 )      F77FF6
32573  LDX ( 136 )        9E88
32575  LDA 124            867C
32577  STA [x]            A784
32579  LEA X 1            3001
32581  STX ( 136 )        9F88
32583  JSR ( 32121 )      BD7D79
32586  LDX 0              8E0000
32589  JSR ( 32059 )      BD7D3B
32592  LDX ( 32765 )      BE7FFD
32596  LDY ( 32757 )      10BE7FF5
32599  STB [x]            E784
32601  LEA X 1            3001
32603  LEA Y 63           313F
32606  CMP Y 0            108C0000
32609  BNE - 12           26F4
32611  JMP ( 32012 )      7E7D0C
32614  LDA 126            867E
32616  LDX ( 136 )        9E88
32618  STA [x]            A784
32620  LEA X 1            3001
32622  STX ( 136 )        9F88
32624  JSR ( 32121 )      BD7D79
32627  LDX 0              8E0000
32630  JSR ( 32059 )      BD7D3B
32633  STB ( 32757 )      F77FF5
32636  LDX 2              8E0002
32639  JSR ( 32059 )      BD7D3B
32642  STB ( 32758 )      F77FF6
32645  LDA 126            867E
32647  LDX ( 136 )        9E88
32649  STA [x]            A784
32651  LEA X 1            3001
32653  STX ( 136 )        9F88
32655  JSR ( 32121 )      BD7D79
32658  LDX 0              8E0000
32661  JSR ( 32059 )      BD7D3B
32664  STB ( 32759 )      F77FF7
32667  LDX 2              8E0002
32670  JSR ( 32059 )      BD7D3B
32673  STB ( 32760 )      F77FF8
32676  LDX ( 32765 )      BE7FFD
32680  LDY ( 32759 )      10BE7FF7
32683  LDA [x]            A6A4
32685  STA [y]            A7A4
32687  LEA X 1            3001
32689  LEA Y 33           3121
32692  STY ( 32759 )      10BF7FF7
32696  LDY ( 32757 )      10BE7FF5
32699  LEA Y 63           313F
32702  STY ( 32757 )      10BF7FF5
32706  CMP Y 0            108C0000
32709  BNE - 32           26E0
32711  JMP ( 32012 )      7E7D0C
```

EVERY ONE A

ORBITER

ALIEN DROPOUT

CYBER RATS

SLIPPERY SID

INVADERS

MASTERPIECE

ASTEROIDS

STARSHIP ENTERPRISE

GROUND ATTACK

.direct to your door

Although the NewBrain is conceived as a total system, the unexpanded Processor itself has a great deal to offer. It is available in two forms: Model AD, shown below, with a built-in line display; and Model A, without the line display. Both models can operate with a monitor or a television set.

MEMORY

- 24K bytes of ROM;
- 32 bytes of RAM, at least 28K of which is available to the user.

THE SCREEN DISPLAY

- 40 or 80 characters to the line – without affecting the 28K bytes of RAM at your disposal;
- 24 or 30 lines to the screen;
- well-formed characters, with true descenders;
- a full European character set;
- normal or reverse video, high resolution graphics on screen of controllable size, 256, 320, 512 or 640 horizontal resolution by 250 vertical lines;
- a facility to set up a "page" of up to 255 lines, with the screen acting as a "window" to display it;
- ability to maintain several such pages simultaneously, and to switch rapidly between them;
- text may be used on graphics screen as well as on parts of the video screen not used by graphics.

CHARACTER SET

- 512 characters, including the full ASCII set, all European accented characters, Greek and graphics symbols.

GRAPHICS

- 20 powerful graphics commands;
- all text characters usable on the graphics screen;
- variable-sized graphics screen, with the rest of the screen available for text – for versatility and to save memory.

SOFTWARE

Enhanced ANSI BASIC; screen editor (32 commands); mathematics package (10 significant figures); graphics commands.

- a very friendly screen editor – a delight to use and readily adapted to text processing;
- arithmetic to 10 significant figures;
- very controllable output formatting of numbers – invaluable for accounting, statistics, and scientific applications;
- a powerful, much enhanced BASIC;
- a very flexible operating system, which allows any data stream to be opened to any device.

INTERFACES

- two tape cassette ports built into the processor unit;
- a built-in printer interface;
- a built-in communications interface (V24/RS232);
- a video monitor interface;
- a TV interface;
- an expansion interface for NewBrain system expansion modules.

KEYBOARD

- standard typewriter pitch, action, layout and size, with editing control and graphics keys.

You can get everything in the box on the

If you understand the facts and figures on the left you'll soon realise that NewBrain has to be one of the most powerful micros around.

However, if you find the box on the left a little hard to follow, don't worry.

We've got over 120 dealers nationwide who've got all the answers at their fingertips.

Either way you'll discover that NewBrain is the kind of micro that will stop the competition getting a look in.

At £269 it starts off with twice as much memory as most of its competitors and can expand to over thirty times that amount. So there's no chance of being left behind in the micro race.

It comes with a very powerful language (enhanced ANSI BASIC) and it'll take CP/M, so it'll work on the same system as similar big business micros, giving you the capacity to use an almost limitless variety of tried and tested software.

But most of all NewBrain is a machine that can expand.

It's designed to take disks, printers and memory expansion modules (up to 2M bytes) plus anything else you'd expect a professional business micro to handle.

So, whether you understand the box on the left or not, pay a visit to someone in the know on the right.

They'll answer all your questions and give you a full demonstration.

New-Brain

Grundy Business Systems Ltd., Grundy House, Somerset Road, Teddington.

left from anyone in the box on the right.

IF YOU ENTERED the programs last month, then you may have been perturbed by the speed of them: how can we slow them down?

The most common method is using a delay subroutine like the one in figure 1.

This is a delay loop. It works by introducing a delay of around 0.5 seconds each time it is called. Unfortunately, it resets the X and Y registers to 0, which in a program where X and Y are used extensively is not a good idea. To get around this we grab two free memory bytes and use them as variables. This makes the program a bit harder to understand, but now X and Y are not changed within the subroutine.

Any two free memory bytes will do. I used $FE and $FF — 254 and 255 — in the example in figure 2 as they are not used on the Vic-20, but any two will do.

We still have the same overall effect, but X and Y remain unchanged. Unfortunately we had to use the accumulator, but this is excusable as 90 percent of all machine-code subroutines use it and so it is unwise to store anything of importance in there for too long a time. So we have a delay loop. Let us add it to our sound program of last month. See figure 3.

Line 100 in last month's loader would now read:

```
100 DATA A9,00,8D,0E,90, A2,FF,A0,00,CA,
    8E,0A,90,8C,0B,90,20,42,1A
102 DATA C8,D0,F3,60,A9,00,85,FE,85,FF,E6,
    FF,D0,FC,E6,FE,D0,F6,60
```

An important point to note is that the branch instruction in the sound section has changed from last month so as to obtain a better effect. As we have added a JSR instruction into our loop, we have added three extra bytes of code, so the branch has altered correspondingly. To run it, type in SYS 6700 and you will experience some weird effects.

This is what we call reading the memory and not remembering the post-code. So far three types of command have been used: first, commands which do not have an operand, for example INY and DEX. Second, branch instructions which precede a displacement rather than an actual memory address. Third, Jump commands — JSR — which precede a memory location. Fourth, commands which access the contents of a specified address.

This is just the tip of the iceberg; there are seven different ways of addressing memory: immediate, absolute, zero page, relative, indirect, indirect pre-indexed with X, and implied-implied addressing.

In immediate addressing the actual value of the operand is manipulated. The immediate addressing is expressed by putting a hash sign in front of the operand. Thus,

LDA # 5

means load the accumulator with 5, or in Basic $A = 5$. With absolute addressing we Load from, Add to or Store into an address which is stored in two bytes immediately after the opcode. So

LDA 2048

means load the accumulator with the value stored in byte 2048, or, in Basic, $A = Peek$ (2048).

Zero-page is absolute addressing except that all the locations are in zero-page — bytes 0 to 255 — inclusive. As one byte can hold values of 0 to 255, the zero-page addressing requires two bytes, op-code and address. In this case,

LDA 38

means load the accumulator with the value stored in byte 0038. The Basic equivalent is simply

$$A = PEEK(3*16 + 8)$$

Relative addressing is the type of addressing used in branch statements, that is, a displacement which is added to or subtracted from the program counter.

BEQ 3A

means if the zero flag is set — $Z = 1$ — in the status register then skip 3A bytes of memory. There is no direct equivalent to this in Basic, but if we imagined that * symbolised the current line of code being executed, then

```
10 REM
20 X = X−1
30 IF X = 0 THEN * = *−10 : REM GOTO 20
```

With the indirect post-indexed with Y addressing technique, the contents of the byte specified in the operand and the following byte are converted into an address. The contents of Y are then added to this address and it is then this address that is manipulated. The indirect with Y is symbolised by brackets around the Operand and then a ",Y" outside the bracket to indicate that it is post- and not pre-indexed; for example:

LDA (32),Y

If Y is equal to 12 decimal, byte 32 is equal to 41 decimal and byte 33 is equal to 22 decimal.

Then the accumulator would be loaded with the contents of byte 5685 — where

$$5685 = 12 + 41 + 22*256$$

In Basic this would appear as

$$A = PEEK (PEEK(32) + 256*PEEK(33) + Y)$$

Indirect — pre-indexed with X addressing is similar to the previous case except now we add the contents of the X register to the operand and then take this new operand as the address of the first byte of the final address acquired. The symbolism for this type of addressing is again the brackets around the Operand but now we put a X inside the brackets:

LDA (32,X)

If X = 4, byte 36 = 201 and byte 37 = 21 then the accumulator would be loaded with the contents of byte 5577, where

$$5577 = 201 + 21*256$$

In Basic this would look something like
$$A = PEEK(PEEK(32 + x) + 256*PEEK(32 + x))$$

Implied-implied addressing requires no operand and often has very little to do with the memory at all. The op-codes in this category are mainly for changing the data in the registers. Thus INY means increment the Y register:

$$Y = Y + 1$$

If one byte can contain 256 different values, then how can a computer access a memory of 65,536 — 64K — bytes? The answer is to use two bytes to represent each address.

So using two bytes for every address, we can have a total of 256 multiplied by 256 — 64K — different addresses. It is no coincidence, then, that if you add up the total amount of RAM memory and ROM memory you get 64K. Some computers such as the CBM-64 use what is known as a virtual memory system whereby the computer splits the memory up into lots of 16K blocks and then pretends to only have 16K at a time.

An address is made up of two bytes: Hi-byte — or page number, where one page is 256 bytes and 256 pages is 64K — and another byte to represent one particular byte out of the 256 in one page, the Lo-byte.

Life is made simpler if hexadecimal is used throughout, as by definition in a hex number such as 1234. The page number is 12 leaving the 34 as the Lo-byte or byte number. The best way to convert a decimal number into a machine-code address — if the computer is at hand — is to go through the following sequence. For example, 10845 decimal is equal to.

```
PRINT INT(10845/4096)  = 2      = 2
PRINT (10845–2*4096)   = 2653
PRINT INT(2653/256)    = 10     = A
PRINT (2653–10*256)    = 93
PRINT (93/16)          = 5      = 5
PRINT (93–5*16)        = 13     = D = 2A5D hex
```

The convention for storing addresses in machine code is that they should always be in a Lo-byte, Hi-byte order. So 10845 would be stored as 5D,2A and not as 2A,5D.

A program with all two-byte addresses will be slower to run than one containing only one-byte addresses, but having only one byte to represent an address means a maximum 255 — one page — of memory. To speed things up, we take the first 256 bytes of memory — 0000 00FF — and call them the zero-page — due to the page number being equal to zero.

A lot of both business and games programs use machine code to move through memory one byte at a time. Word processors have a cursor rushing through sheets of text and

Space Invader games have a running total of the number of bullets on the screen at a time. To do this in Basic, we would have something to the tune of

```
FORA=0TON : POKEA,X : NEXT
```

Notice the use of A as an address which varies. This is an indirect Poke; load X into the byte whose address is stored in the variable A; to convert this into machine code it would be best then to use an indirect command pre- or post-indexed with either X or Y respectively.
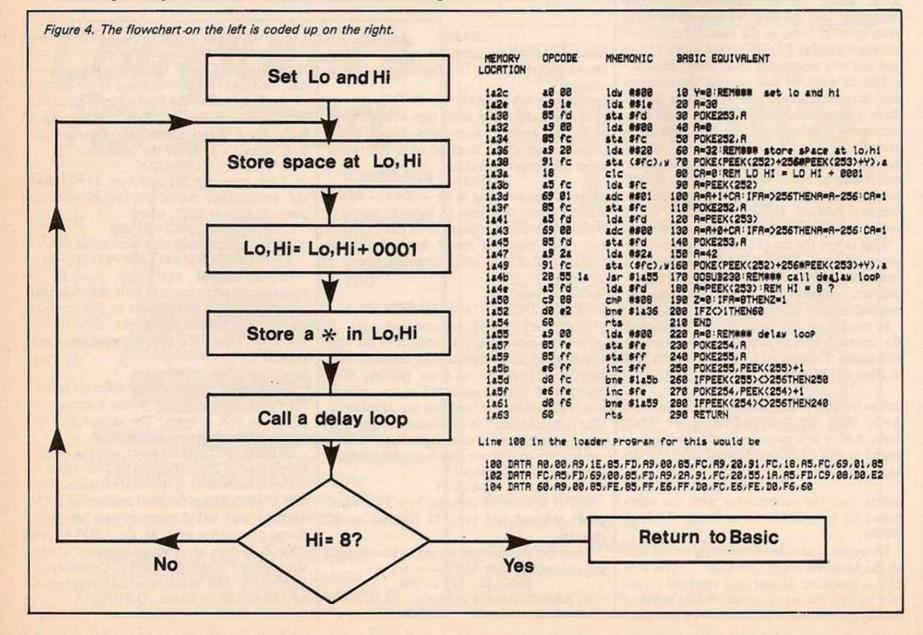
Try moving a character across and down the screen on a Vic. The screen is larger than 256 bytes so we are going to have to use a Lo-byte Hi-byte system and increment it as we move our * across. But first, advanced adding up should be considered.
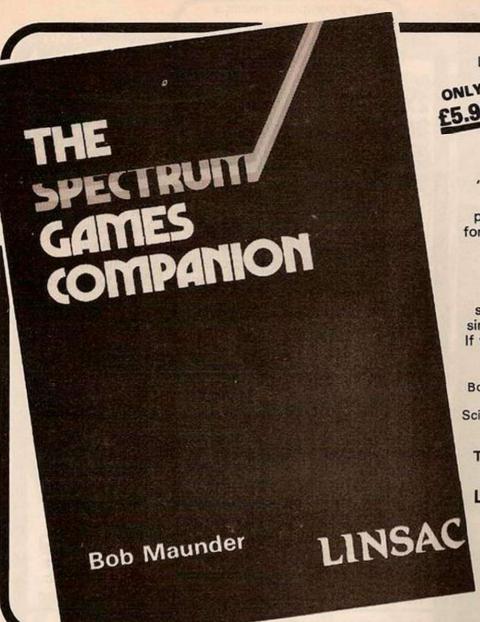
When adding up in two bytes we can still use the ADC — Add with carry — command which can be thought of as

ADC : A = A + (data) + (carry)

Add the data represented in the ADC command to the contents of the accumulator and then add the contents of the carry flag — either 0 or 1.

To add in two bytes we add the Lo-bytes, store them and then add the Hi-bytes. Simple? Not quite. We need some way of telling whether the total of the two Lo-bytes came to more than 255 so that we can either add an extra 255 in the form of incrementing the Hi-byte total or not as the case may be. Enter the carry flag. The adding of the carry flag, which at first seemed to be a nuisance now makes sense in that in a two-byte addition, if the sum of Lo-bytes is greater than 255 this is automatically taken care of in the Hi-byte stage thanks to the ADC command adding the carry flag for us. A flowchart can now be translated into machine code.

Figure 1.

OPCODE	MNEMONIC	BASIC EQUIVALENT
A2 00	LDX #0	10 X=0
A0 00	LDY #0	20 Y=0
C8	INY	30 Y=Y+1
D0 FD	BNE FD	40 IF Y<> 256 THEN 30
E8	INX	50 X=X+1
D0 F8	BNE F8	60 IF X<> 256 THEN 20
60	RTS	70 RETURN

Figure 2.

OPERAND	MNEMONIC	BASIC EQUIVALENT
A9 00	LDA #0	10 A=0
85 FE	STA FE	20 POKE254,A
85 FF	STA FF	30 POKE255,A
E6 FF	INC FF	40 POKE255,PEEK(255+1)
D0 FC	BNE FC	50 IFPEEK(255)<>0THEN40
E6 FE	INC FE	60 POKE254,(PEEK(254)+1)
D0 F6	BNE F6	70 PEEK(254)<70THEN30
60	RTS	80 RETURN

Figure 3.

MEMORY LOCATION	OPCODE	MNEMONIC	BASIC
1A2C	A9 00	LDA #0F	10 A=15
1A2E	8D 0E 90	STA 900E	20 POKE 36878,A
1A31	A2 FF	LDX #FF	30 X=255
1A33	A0 00	LDY #0	40 Y=0
1A35	CA	DEX	50 X=X-1
1A36	8E 0A 90	STX 900A	60 POKE 36874,X
1A39	8C 0B 90	STY 900B	70 POKE 36875,X
1A3C	20 42 1A	JSR 1A42	80 GOSUB 120
1A3F	C8	INY	90 Y=Y+1
1A40	D0 F3	BNE 1A34	100 IF Y<256THEN 50
1A42	60	RTS	110 END
1A43	A9 00	LDA #0	120 REM
1A45	85 FE	STA FE	130 REM
1A47	85 FF	STA FF	140 REM
1A49	E6 FF	INC FF	150 REM
1A4B	D0 FC	BNE 1A48	160 REM
1A4D	E6 FE	INC FE	170 REM
1A4F	D0 F6	BNE 1A47	180 REM
1A51	60	RTS	190 REM

Figure 4. The flowchart on the left is coded up on the right.

MEMORY LOCATION	OPCODE	MNEMONIC	BASIC EQUIVALENT
1a2c	a0 00	ldy #$00	10 Y=0:REM### set lo and hi
1a2e	a9 1e	lda #$1e	20 A=30
1a30	85 fd	sta $fd	30 POKE253,A
1a32	a9 00	lda #$00	40 A=0
1a34	85 fc	sta $fc	50 POKE252,A
1a36	a9 20	lda #$20	60 A=32:REM### store space at lo,hi
1a38	91 fc	sta ($fc),y	70 POKE(PEEK(252)+256*PEEK(253)+Y),a
1a3a	18	clc	80 CA=0:REM LO HI = LO HI + 0001
1a3b	a5 fc	lda $fc	90 A=PEEK(252)
1a3d	69 01	adc #$01	100 A=A+1+CA:IFA=>256THENA=A-256:CA=1
1a3f	85 fc	sta $fc	110 POKE252,A
1a41	a5 fd	lda $fd	120 A=PEEK(253)
1a43	69 00	adc #$00	130 A=A+0+CA:IFA=>256THENA=A-256:CA=1
1a45	85 fd	sta $fd	140 POKE253,A
1a47	a9 2a	lda #$2a	150 A=42
1a49	91 fc	sta ($fc),y	160 POKE(PEEK(252)+256*PEEK(253)+Y),a
1a4b	20 55 1a	jsr $1a55	170 GOSUB230:REM### call delay loop
1a4e	a5 fd	lda $fd	180 A=PEEK(253):REM HI = 8 ?
1a50	c9 08	cmp #$08	190 Z=0:IFA=8THENZ=1
1a52	d0 e2	bne $1a36	200 IFZ<>1THEN60
1a54	60	rts	210 END
1a55	a9 00	lda #$00	220 A=0:REM### delay loop
1a57	85 fe	sta $fe	230 POKE254,A
1a59	85 ff	sta $ff	240 POKE255,A
1a5b	e6 ff	inc $ff	250 POKE255,PEEK(255)+1
1a5d	d0 fc	bne $1a5b	260 IFPEEK(255)<>256THEN250
1a5f	e6 fe	inc $fe	270 POKE254,PEEK(254)+1
1a61	d0 f6	bne $1a59	280 IFPEEK(254)<>256THEN240
1a63	60	rts	290 RETURN

Line 100 in the loader program for this would be

```
100 DATA A0,00,A9,1E,85,FD,A9,00,85,FC,A9,20,91,FC,18,A5,FC,69,01,85
102 DATA FC,A5,FD,69,00,85,FD,A9,2A,91,FC,20,55,1A,A5,FD,C9,08,D0,E2
104 DATA 60,A9,00,85,FE,85,FF,E6,FF,D0,FC,E6,FE,D0,F6,60
```

If you find that setting the parameters of a machine-code routine is a bit of a drag, Stewart Nicholls' program could give you the best of both worlds.

SPECTRUM

NO BASIC SORTING VARIABLES

DO YOU GET tired of having to set parameters of a machine-code routine by Poking numerous addresses with the necessary values? If you do, the following machine-code routine is the answer: it makes Poking completely unnecessary.

The problem is avoided by setting up the values using Basic variables and then searching the variables area to find their value and storing them in the spare bytes of the printer buffer.

To demonstrate the technique, I have attached the search program to a routine which prints a message anywhere on the screen using characters of any height or width, in any ink colour.

The x and y variables are used for the start of the message; since these are the co-ordinates of the top-left corner of the first character of the message the usual Spectrum plot positions apply. That is, 0,0 is the bottom left-hand corner of the screen. For the height of the characters, h is used. Normal height is 1, 22 will be full screen. Width of characters is w. Normal width is 1, 32 is full screen for one character.

The string containing the message to be printed will be a$. The only limitations on the message are that it can only contain characters from code 32 — space — to code 127 — copyright — and that the number of characters multiplied by width of character should not exceed 32 with the Plot position at the left-hand edge of screen. If this is exceeded the program will not crash but the message will wrap round the screen overprinting as it goes.

The program is fully mug-trapped and any wrong parameter of x and y will be corrected before plotting the string. The machine code to find the variables x, y, h, w and a$ is shown in listing 1.

If you read through Chapter 24 of the Sinclair Manual you will see that there are six types of variable, namely: a number whose name is one letter — type (i); a number whose name is two or more letters — type (ii); an array of numbers — type (iii); For-Next loop control — type (iv); string variable — type (v); an array of characters — type (vi).

Each of these variables is uniquely identified by its first byte.

Variable type i = 96d + (letter code – 96d)
Variable type ii = 160d + (first letter code – 96d)

ENTER THE DRAGON

TEACH YOUR DRAGON TO ROAR!

A fresh and highly creative collection of ready made programs designed specifically for the **Dragon 32** micro. Created by the sharpest minds in micro software today, this book will teach you how to maximise the entire range of your Dragon's impressive capabilities. Using easy-to-follow program listings, this book turns your Dragon 32 into a complete arcade of fast action space and adventure games. All your favourites are there including: **Dragon Invaders, Lunar Lander** and **Astral Storm,** plus there's an entire collection of fresh and exciting new games!

Each program fully extends the excellent colour resolution of the Dragon to its maximum and the action is faster than you ever thought possible.

Your overall knowledge of computer operation will be expanded too. Each program features an in-depth explanation of how and why it runs and illustrations of screen displays back up the text to ensure you understand exactly what will be achieved.

The book even includes many programming tips and hints usually reserved only for the experts. So whether you are a first-time computer user or an 'old hand' if you want to make your Dragon really perform, and feel the satisfaction of doing it yourself, this is the book that will show you how!

Variable type iii = 128d + (letter code – 96d)
Variable type iv = 224d + (letter code – 96d)
Variable type v = 64d + (letter code – 96d)
Variable type vi = 192d + (letter code – 96d)
The variables we will be searching for will be types (i) and (v).

If we take, for example, the variable x which has a character code of 120, then using the formula (i) given, its unique first byte is 120. So in order to find variable x we must search through the variables for this unique code, which will then point to the start of the x variable information. This is not just a simple case of running through the variables one byte at a time using a CPIR instruction, as this may find the code 120 held in a string or an array of characters. So we must find the start of each variable, check if it is the one we want, and if not, then jump to the start of the next variable. This means that we must identify the type of variable found before we know how far to jump to the next one. This is not as difficult a task as it might first appear because of the way in which the Spectrum stores the variable parameters.

With three of the variables, namely array of numbers, array of characters and strings, the length of the variable is held in the two bytes following the unique code byte. What is more helpful, bit 5 of the unique code is zero and the remaining three variables have bit 5 set at one

The length of the three remaining variables can be calculated by this method:
Type (i) length = 6 bytes including unique code.
Type (ii) length = number of letters of variable + 5 bytes
Type (iv) length = 19 bytes including unique code.

We now have the information to enable us to jump over each type of variable, and the checks must be made in the following order: first, check if code is 120; if yes then return from routine; check bit 5 and if zero jump by
    value in next two bytes + 1 : GOTO 1
Check bit 6 and if zero check following bytes for

        BIT 7 = 1

that is the last character of the variable, then jump six bytes and Goto 1. Check bit 7 and if zero then jump 6 bytes and Goto 1. Now it must be a For/Next loop so jump 19 bytes and Goto 1.

The mnemonics of the machine-code program to do this are shown in the Find subroutine. This uses address 23728 to hold the unique code of the variable for which we are searching. The HL register holds the address being checked, and the accumulator is loaded with the unique code from address 23728.

The subroutine only takes 46 bytes to check, Jump and locate the start address of any variable.

Now that we have found the start address of our variable x we must find its value. You will see that the value for whole numbers is held in the third and fourth bytes following the unique code. In our case we are only interested in numbers from 0 to 255 for x. So we can ignore the high byte and store the low byte, that is the third byte. This applies to all our variables x, y, h and w.

With this in mind a subroutine, Setup, can

be assembled to move along three bytes from the address held in HL and then store the value held in this address in the printer buffer area of memory. So we now have the means of finding and storing values of x, y, h and w in addresses 23296/7/8 and 9.

We can now move to the slightly more complicated string parameters. In our case we need to find a$, unique code 65. The same Find routine can be used to find the start of the variable. Once found we then move to the next two bytes to find the LEN of the string as follows:

```
INC HL
LD E,(HL)
INC HL
LD D,(HL)
```

This will put the length of the string into the DE register, and, as we can safely limit the length of the string to 255 characters we only need to store the number in low byte:

```
LD (23300),DE
```

The number in address 23301 will be overwritten with the first character in the string:

```
PUSH DE     Get number of characters into
            BC
POP BC
INC HL      Set HI at start of characters
LD  DE,23301
LDIR        Transfer information
```

The above will transfer the string characters from the variables to the printer buffer starting at address 23301.

With this routine we now have x, y, h, w, length of string, and characters in string stored consecutively in the printer buffer, and can go straight into the routine to plot the string based on these parameters.

The plot routine finds the start of the eight bytes for each character in the character set and, for each byte, a Rotate Left instruction is carried out either plotting or unplotting depending whether Carry is set or not. Again use is made of the two unused bytes in the Systems Variables 23728/9 to hold and update the x,y plot positions for each character.

If the y plot position goes below zero, then it is reset to 175 to give a wrap around effect and likewise if the x position goes above 255 it is reset to zero. So now we have a machine-code program that can be called from Basic with no Pokes in sight — for example:

```
10 LET x = 0 : LET y = 100 : LET h = 8 :
   LET w = 4 : LET a$ = "Finished" : INK 6 :
   RANDOMISE USR 32393
```

This saves 13 Pokes including LEN a$. Listing 1 shows the Basic program to set up the machine code above RAMtop. On a 16K Spectrum this is immediately before the user-defined graphics and occupies 255 bytes. Once the machine-code is entered the Basic can be Newed and the code saved in the usual way.

There are a couple of points to watch with this program. First, ensure that all the variables are defined before a call is made to the routine as failure to find a variable will crash the program. Second, Do not use x, y, h, or w as a control variable for a For-Next loop because the Spectrum will then delete the simple variable and use the For-Next loop to hold further values of x, y, h and w. This can be demonstrated by the simple Basic program:

```
10 FOR a = 1 TO 10
20 PRINT a ;
```

```
30 NEXT a
40 LET a = 3
50 NEXT a
```

This program will produce a continuous loop resetting a to 3 and jumping back into the For-Next loop showing that the variable — a — in line 40 is part of the For-Next loop control variable. If this was not the case the Spectrum would give an error report:

    1 NEXT without FOR, 50:1

You can have a small Basic subroutine to work out the width of characters and the start plot position to give information printed centrally on any line; such a program is given in listing 2 with sample printout.

We can now also tackle a machine-code program to print out all the variables used after a program has been Run. Note that this may not be all the variables in the listing as when a program is Run certain subroutines may not have been called and so variables held in those routines will not have been placed into the variables area.

This program — listing 3 — could be of use in debugging Basic programs. Listing 3 is in the form of a machine-code dump which should be held in Data statements as listing 1, and Poked into a memory position of your choice. There are no Jump or Call commands to within the program. I find it handy to have two versions of the program one stored above RAMtop and another in a line 1 Rem statement containing 311 zeros and called using

        RANDOMISE USR 23760

The machine-code is then held in the Basic area of RAM and can be Merged with any program — as long as the program does not have a line 1, which will be overwritten. One point to note when Saving the Basic Rem statement on tape is to use Clear to erase all variables used in the machine-code loader, otherwise these will be saved and Merged along with the Rem statement. This is good practice on any program which does not require the variables to be saved.

Listing 4 is a demonstration program setting up variables, and shows a screen copy of the machine-code output which lists these variables under their variable types. Notice how the Spectrum converts all variables to lower case and also that variables a, h, l, o, w, x and z have been listed under the For-Next loop only and not under number.

Listing 5 shows the machine-code mnemonics for assembly into a Rem statement, for anyone who wishes to check through the program to see how it works. The first section from address 5CD0 to 5CE5 sets up the Border, Paper and Ink colours; from 5CE6 to 5D46 prints the heading. The remainder of the program checks the variables one by one finding the type of variable, locating the next Print position in that variables column, updating the Print position and printing the variable.

The section from 5D94 to 5DBE is interesting in that it checks for a column reaching the bottom of the screen and if it does then waits for a key press before scrolling 21 lines, leaving the heading on the screen and printing the next variable in position. A return to Basic is only made when all the variables have been listed.

(continued from page 137)

Start address for 16K machine code 32235.
ORG 32335
Subroutine to find variables.

Routine to plot the string.

```
                                        LD    HL,(23296)           END     POP   AF
         LD    HL,(23627)         XOR   A                            POP   BC
START    LD    A,(23728)          LD    A,H                          DJNZ  LOOP2
         CP    (HL)               SBC   176                   LD    A,(23296)
         RET   Z                  JR    C,YES                        LD    HL,23728
         BIT   5,(HL)             LD    H,A                          LD    (HL),A
         JR    NZ,NEXT      YES   LD    (23296),HL                   INC   HL
         INC   HL                 LD    (23728),HL                   XOR   A
         LD    E,(HL)       RUN1  PUSH  HL                           LD    A,(HL)
         INC   HL                 LD    A,(HL)                       SBC   176
         LD    D,(HL)             LD    H,0                          JR    C,OK
         ADD   HL,DE              LD    L,A                          LD    (HL),A
         INC   HL                 ADD   HL,HL                        JR    CONT
         JR    START              ADD   HL,HL            OK    LD    A,(HL)
NEXT     BIT   6,(HL)             ADD   HL,HL                        CP    0
         JR    NZ,NEXT1           LD    DE,15360                     JR    NZ,OK1
LOOP5    INC   HL                 ADD   HL,DE                        LD    (HL),176
         LD    A,(HL)             LD    B,6              OK1   DEC   (HL)
         BIT   7,A          LOOP4 PUSH  BC               CONT  POP   BC
         JR    Z,LOOP5            LD    BC,(23297)                   POP   HL
LOOP6    LD    DE,6         LOOP3 LD    A,(HL)                       DJNZ  LOOP3
         ADD   HL,DE              PUSH  HL                           INC   HL
         JR    START              PUSH  BC                           POP   BC
NEXT1    BIT   7,(HL)             LD    B,8                          DJNZ  LOOP4
         JR    Z,LOOP6      LOOP2 PUSH  BC                     LD    A,(23299)
         LD    DE,19              RLA                            ADD   A
         ADD   HL,DE              PUSH  AF                       ADD   A
         JR    START              JP    C,PLOT                   ADD   A
                                  LD    HL,(23299)               LD    L,A
                                  LD    A,(23728)                LD    A,(23728)
Subroutine to set parameters      LD    (23728),A                ADD   L
of variable to be found.          JP    END                      LD    (23296),A
                                  PLOT  LD    BC,(23298)          LD    (23728),A
         LD    (23728),A    LOOP1 PUSH  BC                        LD    A,(23297)
         CALL                     LD    BC,(23728)                LD    (23729),A
         INC   HL                 PUSH  BC                        POP   HL
         INC   HL                 CALL  22E5H                     INC   HL
         INC   HL                 POP   BC                        LD    A,(23300)
         LD    A,(HL)             INC   C                         DEC   A
         LD    (BC),A             LD    (23728),BC                RET   Z
         RET                      POP   BC                        LD    (23300),A
                                  DJNZ  LOOP1                     JP    RUN1
                                                                  END
```

Start of machine-code routine.

```
         LD    BC,23296                    LD    (23728),A
         LD    A,120              CALL
         CALL                    INC   HL
         INC   BC                LD    E,(HL)
         LD    A,121             INC   HL
         CALL                    LD    D,(HL)
         INC   BC                LD    (23300),DE
         LD    A,104             PUSH  DE
         CALL                    POP   BC
         INC   BC                INC   HL
         LD    A,119             LD    DE,23301
         CALL                    LDIR
         LD    A,65
```

Example program — listing 2.

```
  10 LET x=0: LET y=175
  20 LET h=4
  25 FOR l=1 TO 5
  30 READ a$
  40 LET w=INT (32/LEN a$)
  50 LET x=INT ((256-(8*w*LEN a$
))/2)
  60 RANDOMIZE USR 32393
  70 LET y=y-8*h
  80 NEXT l
  85 LET y=y+24
  86 READ a$: RANDOMIZE USR 3239
3: STOP
  90 DATA "Program:PLOT","©1983
S.Nicholls","******************
***************","YOUR COMPUTER","
APRIL 1983","
"
```

Listing 1.

```
   1 REM x=x axis
   2 REM y=y axis
   3 REM h=height
   4 REM w=width
   5 REM a$=message
  10 CLEAR 32334
  20 LET start=32335
  30 FOR a=start TO 32599
  40 READ code
  50 POKE a,code
  60 NEXT a
  70 DATA 42,75,92,58,176,92,190
,200,203,110,32,8,35,94,35,86,25
,35,24,239,203,118,32,12
  80 DATA 35,126,203,127,40,250,
17,6,0,25,24,223,203,126,40,246,
17,19,0,25,24,213,50,176
  90 DATA 92,205,79,125,35,35,35
,126,2,201,1,0,91,62,120,205,125
,126,3,62,121,205,125,126
 100 DATA 3,62,104,205,125,126,3
,62,119,205,125,126,62,65,50,176
,92,205,79,126,35,94,35,86
 110 DATA 237,83,4,91,213,193,35
```

```
,17,5,91,237,176,42,0,91,175,124
,222,176,56,4,103,34,0
 120 DATA 91,34,175,92,35,5,91,2
29,126,38,0,111,41,41,41,17,0,60
,25,6,8,197,237,75
 130 DATA 1,91,126,229,197,6,8,1
97,23,245,218,249,126,42,3,91
 140 DATA 58,176,92,133,50,176,9
2,195,15,127,237,75,2,91,197,237
,75,176,92,197,205,229,34,193
 150 DATA 12,237,67,176,92,193,1
6,239,241,193,16,211,58,0,91,35,
176,92,119,35,175,126,222,176
 160 DATA 56,3,119,24,8,126,254,
0,32,2,54,176,53,193,225,16,177,
35,193,16,168,58,3,91,135,135,13
5,111,58,176,92,133
 170 DATA 50,0,91,50,176,92,58,1
91,50,177,92,225,35,58,4,91,61,
200,50,4,91,195,206,126
 200 LET x=0: LET y=100: LET h=8
: LET w=4: LET a$="Finished"
 210 RANDOMIZE USR 32393
 220 STOP
9999 SAVE "large" LINE 1
```

Listing 4.
```
  1 FOR a=1 TO 2: NEXT a
  2 DIM b$(7)
  3 LET c$="#"
  4 DIM d(7)
  5 LET e=1
  6 LET G$(a)(s=1
  7 LET A=1
  8 LET B=1
  9 LET C=1
 10 LET D=1
 11 LET E=1
 12 LET F=1
 13 LET G=1
 14 LET H=1
 15 LET I=1
 16 LET J=1
 17 LET K=1
 18 LET L=1
 19 LET M=1
 20 LET N=1
 21 LET O=1
 22 LET P=1
 23 LET Q=1
 24 LET R=1
 25 LET S=1
 26 LET T=1
 27 LET U=1
 28 LET V=1
 29 LET W=1
 30 LET X=1
 31 LET Y=1
 32 LET Z=1
 33 LET D$=";"
 34 LET F$="2"
 35 LET DEFINE=3
 36 FOR L=1 TO 10: NEXT L
 37 LET data=2
 38 LET origin=3
 39 LET print=1
 40 LET help=1
 50 LET a$="e"
 51 LET g$="o"
 52 DIM v$(8)
 53 FOR h=1 TO 2: NEXT h
 54 FOR w=34 TO 35: NEXT w
 55 LET d$="#"
 56 LET a$="1"
 57 FOR x=1 TO 4: NEXT x
 58 FOR z=1 TO 5: NEXT z
 60 DIM s(1)
 61 DIM f(7)
 62 DIM j$(8,7)
 63 DIM b(2)
 64 FOR o=1 TO 2: NEXT o
 86 LPRINT "DEMO BASIC PROGRAM"
 87 PRINT ''
 88 LLIST
500 LPRINT '''"SCREEN COPY OF A
COPY"
510 RANDOMIZE USR 31900
515 LPRINT
550 COPY
550 LPRINT '''"MACHINE CODE DUM
P"
560 LPRINT '''"MAY BE LOCATED AN
YWHERE IN     AVAILABLE MEMORY"
570 LPRINT
600 FOR a=31900 TO 32210 STEP 8
610 FOR b=0 TO 7
620 LPRINT TAB b*4; PEEK (a+b)
630 NEXT b: NEXT a: STOP
9995 CLEAR 31899: LOAD ""CODE :
GO TO 1
9999 SAVE "basicvars" LINE 9998:
SAVE "variables"CODE 51900,312
```

Screen copy of listing 4.
```
FN# $()#$#NO.().#NO.#(WORD) NO.

   a      c   d       e    details
   h|o    v|j f|a  g|d      define
          z                data
                           origin
                           print
                  ijklmnpqrstuvw   help
```

Listing 3.
```
  62    1  205 155  34  253  54  83
  15   62  205   1   22   62 205 107
  13   62   205   1   22   62   22
 215  175 215  175  215  62   78  215
  62   78 215   62  143 215  62   32
  62  143 215   62  143 215  62   79
 215   62  46 215   62  40 215  62
  41  215  62 143 215  62  78 215
  62  145 215   62  62 215  62  79
 215   62  62 215  62  62 215  62
 215   62  62 215  62  87 215  62
  79  215  62  62 215  62  68 215
  62   41 215  62  32 215  62  78
 215   62  79 215  62  46 215   6
  16  352  6    4  175  33    0  41
 119   35   35  198    4  16 249  42
  75   92   34  176  92   17   2  91
  40   30   19  254  64   19 254 192
  19   19  254  64   19  19  19  19
 254  128   40   24  19  19 254  96
  40    5   19    0  24   2  24 210
 237   83  213  91  203    1 174   91
  40  249  203  203  34    1  91
 205   14    6    6  33    2  91
  35  126  254   1  32    1  52  35
 230   31  198  225  24  209 126
   1   32   62   32 215  43   91
 126   52  215  35 126 215 241 215
  42  176  92  203 110  32   6  35
  94   35   85   25  35  24 187 203
  35   32   16  203  226 127 202
   6  215   24  24  145 203 126  40
  17   19    0   25  24  135  62
```

Listing 5.
```
5CD0 3E01      LD   A,01
5CD2 CD9B22    CALL 229B
5CD5 FD36530F  LD   (IY+53),0F
5CD9 3E02      LD   A,02
5CDB CD0116    CALL 1601
5CDE CD0B0D    CALL 0D0B
5CE1 3E02      LD   A,02
5CE3 CD0116    CALL 1601
5CE6 PRINT "FN# $()#$#NO.()#NO.#
(WORD) NO."
5D47 060E      LD   B,0E
5D49 3E02      LD   A,02
5D4B 21005B    LD   HL,5B00
5D4F 77        LD   (HL),A
5D50 23        INC  HL
5D51 10FC      DJNZ 5D4E
5D53 0606      LD   B,06
5D55 AF        XOR  A
5D56 21035B    LD   HL,5B03
5D59 77        LD   (HL),A
5D5A 23        INC  HL
5D5B C604      ADD  A,04
5D5D 10F6      DJNZ 5D55
5D5F 2A455C    LD   HL,(5C55)
5D62 22B55C    LD   (5C60),HL
5D65 11025B    LD   DE,5B02
5D68 7E        LD   A,(HL)
5D69 C8        RET  Z
5D6A E8E0      AND  E0
5D6C E8E0      AND  E0
5D6E FEE0      CP   E0
5D70 281E      JR   Z,5D90
5D72 13        INC  DE
5D73 13        INC  DE
5D74 FEC0      CP   C0
5D76 2818      JR   Z,5D90
5D78 13        INC  DE
5D79 13        INC  DE
5D7A FE40      CP   40
5D7C 2812      JR   Z,5D90
5D7E 13        INC  DE
5D7F 13        INC  DE
5D80 FE80      CP   80
5D82 280C      JR   Z,5D90
5D84 13        INC  DE
5D85 13        INC  DE
5D86 FE60      CP   60
5D88 2806      JR   Z,5D90
5D8A 13        INC  DE
5D8B 13        INC  DE
5D8C 1802      JR   5D90
5D8E 1802      JR   5D90
5D90 ED53005B  LD   (5B00),DE
5D94 4A        LD   A,(DE)
5D95 FE16      CP   16
5D97 2026      JR   NZ,5DBF
5D99 E5        PUSH HL
5D9A D5        PUSH DE
5D9B 76        HALT
5D9C FDCB015E  BIT  5,(IY+01)
5DA0 28F9      JR   Z,5D9B
5DA2 FDCB01AE  RES  5,(IY+01)
5DA6 0E15      LD   B,15
5DA8 CD800E    CALL 0E80
5DAB 0600      LD   B,00
5DAD 21025B    LD   HL,5B02
5DB0 35        DEC  (HL)
5DB1 7E        LD   A,(HL)
5DB2 FE01      CP   01
5DB4 2001      JR   NZ,5DB7
5DB6 34        INC  (HL)
5DB7 23        INC  HL
5DB8 10F5      DJNZ 5DB0
5DBA D1        POP  DE
5DBB E1        POP  HL
5DBC 18D1      JR   5D90
5DBE 7E        LD   A,(HL)
5DC0 E61F      AND  1F
5DC2 C660      ADD  A,60
5DC4 F5        PUSH AF
5DC5 3E02      LD   A,02
5DC7 CD0116    CALL 1601
5DCA 3E16      LD   A,16
5DCC D7        RST  10
5DCD 2A005B    LD   HL,(5B00)
5DD0 7E        LD   A,(HL)
5DD1 34        INC  (HL)
5DD2 D7        RST  10
5DD3 23        INC  HL
5DD4 7E        LD   A,(HL)
5DD5 D7        RST  10
5DD6 F1        POP  HL
5DD7 D7        RST  10
5DD8 2A005C    LD   HL,(5C00)
5DDB CB5E      BIT  3,(HL)
5DDD 2008      JR   NZ,5DE7
5DDF 23        INC  HL
5DE0 5E        LD   E,(HL)
5DE1 23        INC  HL
5DE2 56        LD   D,(HL)
5DE3 19        ADD  HL,DE
5DE4 23        INC  HL
5DE5 18A7      JR   5D8E
5DE7 C076      JR   NZ,5DFD
5DE9 2012      JR   NZ,5DFD
5DEB 23        INC  HL
5DEC 7E        LD   A,(HL)
5DED CB7F      BIT  7,A
5DEF 2003      JR   NZ,5DF4
5DF1 D7        RST  10
5DF2 18F7      JR   5DEB
5DF4 E67F      AND  7F
5DF6 D7        RST  10
5DF7 110600    LD   DE,0006
5DFA 19        ADD  HL,DE
5DFB 1891      JR   5D8E
5DFD CB7E      BIT  7,(HL)
5DFF 28F6      JR   Z,5DF7
5E01 111300    LD   DE,0013
5E04 19        ADD  HL,DE
5E05 1887      JR   5D8E
```

THE IDEAL HOME FOR YOUR HOME COMPUTER

The System Care Desk - made specially for home computer systems. Designed as a neat, smart piece of furniture for your home.

£49·95

Exhibited at the P.C.W. Show and the Northern Computer Fair

And now including design features based on our discussions with home computer users. Send for yours TODAY.
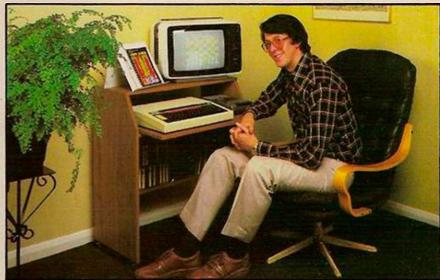
● Supplied as a flat-pack. Easily assembled. Full instructions provided.
● Smart teak effect finish.
● All cables can be neatly and safely 'hidden.'
● Ergonomically designed. Sliding shelf positions the keyboard at the right distance from the screen to suit you. Slides away after use. Saves space.

● Easi-glide castors allow you to move your computer installation where you want it.
● Houses computer, disk drive / cassette recorder and monitor, with ample space for tapes, books etc.
Overall height - 0.83m.
Computer shelf height - 0.70m.
Width - 0.70m.
Depth (with shelf closed) - 0.38m.

ORDER BY FREEPOST - send the coupon or order in writing along with your cheque or postal order in an envelope - no stamp needed. Please allow up to 28 days for delivery. And there's a 14-day money back option, in the event that you're not fully satisfied with your purchases.

4-WAY MINIATURE MASTER PLUG

Provides you with just one 13 amp mains input for up to four 6 amp max outputs to a total loading of 13 amp. Tidy. Safe. Fully insulated.

£8·95

(Only available with orders for the System Care Home Computer Desk).

FREE
A pack of cable tidies with every desk ordered. To ensure safe operation and improve appearance.

3D SPAWN OF EVIL £4.95 Deep into the galactic void, hosts of
green spawn go through a complex multi-stage breeding cycle. Your
task is to seek out and smash the cycle to eliminate the spawn before
it matures to full attack potential. This is the most spectacular 3D space
game ever written for this 16K Spectrum.

METEOROIDS £4.95 The best Spectrum version of this popular
arcade game. Three meteor types which increase and decrease in
frequency. The meteros shatter into many fragments and the
fragments even splatter. Ship drift and decelerate, space walk, and evil
alien ship. This game also contains two character sets.

SOUND FX £4.95 Released by popular demand, this is the program
that was used to create most of the sound effects featured in our
games. It enables you to create an infinitely variable amount of
machine code sounds, and use them in your own basic programs.
Even the beginner can use this program to create fantastic sounds for
his own programs.

DISASSEMBLER £4.95 This is a fast machine code disassembler for
the on the 48K and 16K Spectrum. It gives full colour output to the
screen, grouping together similar instructions in the same colour, IE: all
loading instructions are green. You have options for output into the
screen or printer, this output be in decimal or hex.

FRUIT MACHINE £4.95 An excellent new program from d'k
tronics, with such features as: nudge, hold, gamble. Superb use of
graphics and colour; just like the real thing, and just as addictive !

FROM

dktronics

CREDIT CARD SALES LINE (0799) 22359/26350 (24 hrs).
QUOTING YOUR ACCESS OR BARCLAYCARD NUMBER d'k tronics,
Unit 2, Shire hill Ind. Est., Saffron Walden, Essex. CB11 3AQ.

d'k tronics Software is also available from Boots, WH Smiths and all good Software outlets.

BASIC DICTIONARY

This dictionary, compiled by Tony Edwards, will explain the function of common Basic words as used in popular machines, enabling you to work out your own machine's equivalent. A useful complement to our recent series on Basic dialect translation.

@ (at) Used as an abbreviation for AT. On some computers

                    `<shift> @`

will freeze execution until another key is pressed.

 # (hash) Used as a flag to indicate double precision variables. The BBC Micro uses the hash sign to indicate an immediate operation in assembler. This sign is also used with the meaning "number" when referring to peripherals.

                    `CLOAD # 1`

will load from peripheral device number 1. Also used to represent "not equal to".

$ (dollar) An ANSI standard symbol indicating string variable. It is often used to indicate string functions also.

! (exclamation mark) Used as a flag to indicate single precision variables. Its use is usually optional as variables default to single precision. Some interpreters use the exclamation mark as an abbreviation for Remark.

% (per cent) A flag used to indicate integer variables. It is also used in connection with a Print Using statement to indicate the output is unprintable.

? (question) A common abbreviation for PRINT.

/ (back slash) An operator used to allow multiple statements on a single line. It is also used as an erase indicator when correcting typing errors.

* (asterisk) An alternative for the multiplication operator. It is also used on some computers as the logical AND operator. The BBC Micro uses it to express the binary multiplication of integers and reals.

** (double asterisk) An alternative for the exponentiation operator.

(circumflex) An exponentiation operator

+ (plus) The arithmetical addition operator. Some computers use this symbol in place of the logical OR so care must be exercised with such statements as "IF (A=0) + (B=0)" which will be satisfied if either A or B is equal to 0.

This symbol is also used to indicate concatenation of strings.

- (minus) The arithmetic subtraction operator, also used as a negation operator.

/ (slash) The ANSI standard division operator.

& (ampersand) Widely used as a suffix to indicate hexadecimal numerals.

~ (tilde) Used on the BBC Micro at the start of a print field indicating that the item is to be printed in hexadecimal.

£ (pound) Not usually available, but when it occurs it often replaces the $ and is used in the same way.

↓ (down arrow) Represents a line feed.

→ (right arrow) Moves cursor right.

← (left arrow) Moves cursor left. Also known as "Back arrow".

                    `< shift > ←`

Deletes the current line on some machines.

π (pi) A function which returns the value of PI.

TOWER OF BABEL

Computers do not share a common tongue, because some languages must be task-specific. John Dawson traces their development.

HAVING TRACED some of the history of software from the first faltering machine-code steps in Manchester and Cambridge through the development of Fortran and Forth, this month we look at how information may be regarded as an asset to be examined and manipulated in many ways, rather than as data simply to be fed through a program in order to achieve a pre-ordained printout.

Language tree

First, look at figure 1 and you will see the rough historical order in which high-level languages arose and how the "original" high-level languages have bred their offspring. There are hundreds of computer languages and figure 1 shows only a very few of the more common. Fortran was written for solving scientific problems while Cobol — Common Business-Oriented Language — was commissioned by the Department of Defense in the United States for general "commercial" use. Between them, these two languages are probably used in the great majority of the world's professional computing applications.

How can that be so? There are more sophisticated languages available that are easier for the programmer to use and software development has moved on considerably from the late 1950s when they were first made available commercially. Many professional computing personnel still use Cobol as the language of choice on new hardware. Why? The answer is simple — Cobol and Fortran work, they are well-understood and that makes the task of developing a program or a suite of programs far easier than it would be if an untried language was introduced. Additionally, there are large numbers of sub-routines and larger segments of programs that can be incorporated into a new piece of software.

The person who taught me anaesthetics said that "a crisis is no time to use a new technique that is unfamiliar. Stick to something ordinary that you understand well". Most software development turns into a crisis at some stage.

At the same time there is no doubt that Fortran is a pretty revolting language to use.

"Structured" programming is nearly impossible using Fortran and one of the major advances in the development of programming languages has been the gradual acceptance of the idea that structure makes programs easier to read, hence less prone to error, and, most important, allowing the programmer a less constrained expression of the steps to the solution of the problem. A program built out of a number of modules will be comparatively easy to debug and can be produced by a team of programmers working to some common standards.

Easy debugging

Modules can be built very easily so that there is a single entry point and a single exit. Once again debugging becomes far easier. Forth allows program development by writing and testing the action of discrete words — modularity carried to a useful and logical conclusion. The Goto instruction, a natural development of the first machine-code programs which would go to one instruction or another depending on the result of a test, permits crossed lines of flow in a program, several entry points into a block of code and many exits. The Goto or Jump instruction is intuitively attractive and potentially highly confusing.

Many modern languages have no Goto instructions relying instead on loops controlled by Do — Until or Repeat — While operations to create structure in the program.

Computer languages can be classified in a number of ways. There are fine differences between some versions of a language and, like human speech, there are the identifiable characteristics that allow some languages to be grouped into families. After the loose "commercial" versus "scientific" distinction

epitomised by Fortran and Cobol, there is a split between interactive and batch-processing languages. An interactive computer language provides facilities that allow the programmer to make changes in the program almost as it is executing. The ordinary interpreted Basic supplied with almost every microcomputer is an interactive language. A batch-processing language makes no provision for the operator to modify the program at the computer.

I remember going to a hut behind University College in the late 1960s, sitting at a large card punch to produce a set of cards holding a Fortran program and then handing the cards across a counter in the computer department. The cards also made splendidly original party invitations with a small line of type at the top and coded holes all over the rest of the card. However, it was 24 hours later before I was able to return to collect the results of my program execution.

Fortran has now been implemented, that is installed and made to work on a number of microcomputers and has become more interactive. It is still a compiled language, which means that you must first write the source code, then compile that to object code — machine-code instructions that will execute on your machine — and finally you can run the object code to carry out the operations you wish. If an error occurs you must load the source code back into your computer, find and correct the error and start the process all over again.

Functional languages

Lisp and Prolog are examples of a group of computer languages which are known as functional languages. The other languages shown in figure 1 are "imperative" languages. A program written in an imperative language is composed of a series of statements which are obeyed by the computer. Certainly the machine may choose one of two or more conditional branches from time to time but the program is carefully prepared to take account of those possibilities. In other words, the programmer must describe quite precisely how a result is to be computed, rather than concentrating on the result itself. Functional languages work fundamentally by evluating a function and the general form of a program is:

Results = Function (input values)

For example, a question in Prolog is a program. At a very simple level a Prolog programmer might write:
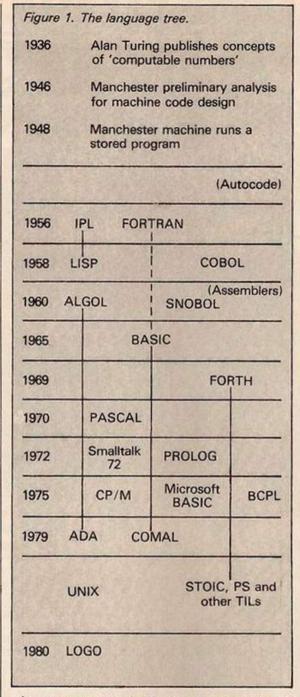
Which (x x sex Male and x illness Glandular Fever)

which might produce this on the screen of the computer:

Answer is (Bertram Leadston)
Answer is (John Salmon)
Answer is (Nigel Mudthorpe)
No (more) answers

if the information had been entered, probably among a lot of other data, under the headings of "name", "sex" and "illness". In relation to the general form of the program. 'Which' is the function to be carried out and the items in the brackets are the input values.

Snobol is an example of a Markov language: that is a language in which the basic operation is to look for a pattern, substitute something in its place and choose the next statement to be carried out on the basis of the match between

Figure 1. The language tree.

Year			
1936	Alan Turing publishes concepts of 'computable numbers'		
1946	Manchester preliminary analysis for machine code design		
1948	Manchester machine runs a stored program		
			(Autocode)
1956	IPL	FORTRAN	
1958	LISP		COBOL
			(Assemblers)
1960	ALGOL	SNOBOL	
1965		BASIC	
1969			FORTH
1970	PASCAL		
1972	Smalltalk 72	PROLOG	
1975	CP/M	Microsoft BASIC	BCPL
1979	ADA	COMAL	
	UNIX		STOIC, PS and other TILs
1980	LOGO		

the two patterns. A Markov statement is said to be a random process in which the probability of a transition to a new state depends only on the current state. D W Barron says in *An Introduction to the Study of Programming Languages* that it is arguable that all algorithms — the sequence of operations necessary to solve a problem — can be reduced ultimately to Markov statements. Snobol 4 is a versatile string-processing language which can be used for text manipulation. You could, for example, look for a match in Snobol by typing the following statements:

STRING "John Dawson" :
S (PRINTOUT) F
(TRYAGAIN)

Data protection

If the words "John Dawson" appear in the characters which make up String — a Successful match — the program will branch to the label Printout. If the match Fails — F — then control will pass to the section of the program labelled Tryagain. There are many other sophisticated search and manipulation instructions in this language and Snobol may be one of the computer languages which could be used in computer applications that particularly worried the Lindop Committee on Data Protection because of the possibility of free text retrieval from very large computer files held about individuals.

The fourth loose distinction between types of computer languages differentiates between ordinary programs such as Basic and Pascal, and real-time computer languages for controlling a process as it occurs.

The hallmark of a real-time language is its ability to respond to external interruptions as peripheral devices generate information that must be taken into consideration by the program.

Because "bit-twiddling" and interrupt handling are difficult to achieve at a high level, most real-time languages tend to work in an intimate relationship with the machine's operating system. Forth, Stoic and the other Threaded Interpretative Languages are able to drop directly into low-level, and hence fast and finely controllable, assembly language which indicates that it should be simpler to try to create a real-time program in Forth than in Basic, Pascal or Fortran.

An aircraft simulator is a good example of an application that may require real-time computing, taking in the responses made by the pilot through the cockpit controls, processing that information, and then changing the attitude, speed and other simulator outputs to match the real aircraft's characteristics.

Sieve race
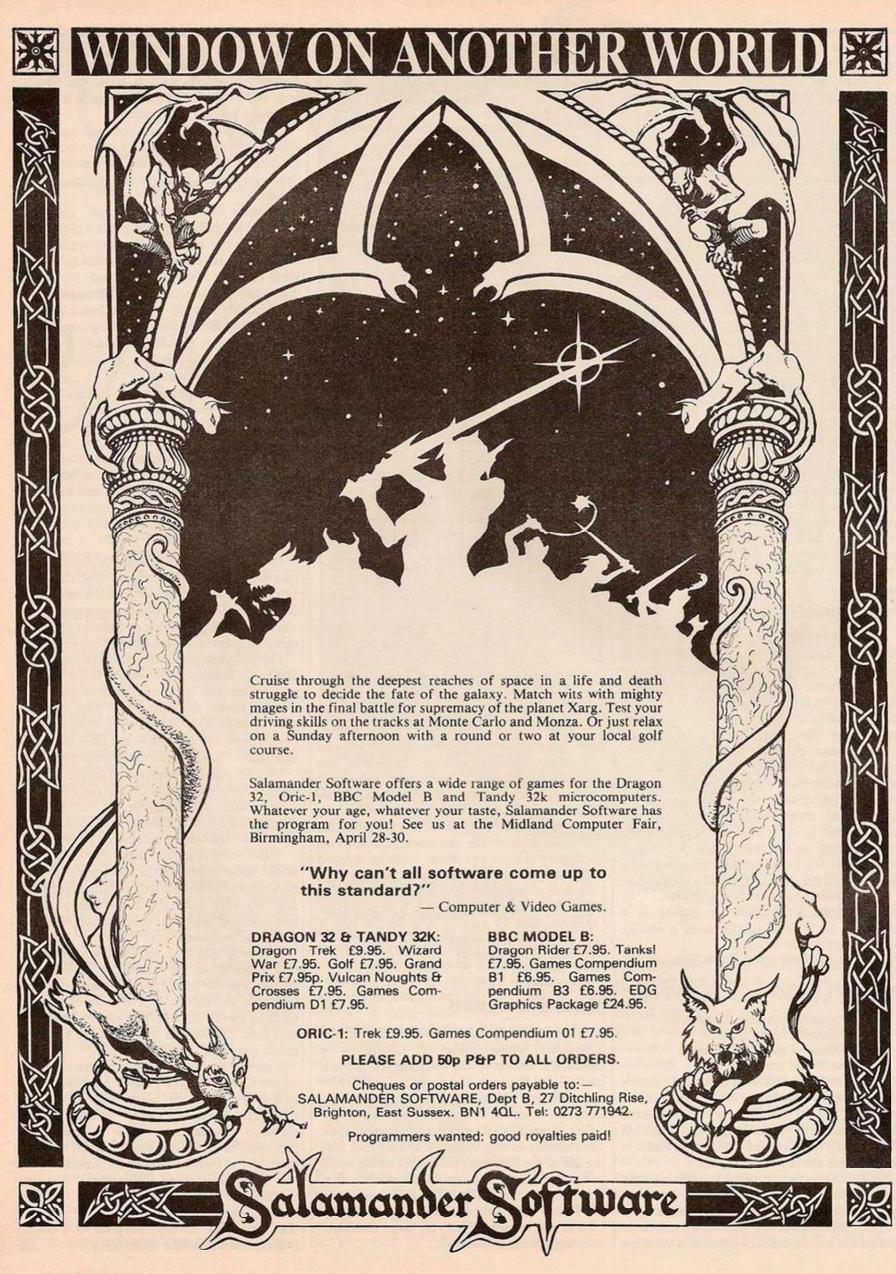
After all the esoteric discussion about which language suits which purpose best there are some simple stark facts based on the results of a January 1983 survey in which the same program was run on a large number of different machines using different languages on different machines.

The program was a standard version of the Sieve of Erastothenes which finds all the prime numbers between 3 and 16,381. The world's fastest computer, the Cray 1, vied with an IBM 3033 for the shortest time using Fortran — 0.110 seconds — and IBM assembly language — 0.0078 seconds — respectively.

Various microcomputers competed for the slowest time which was finally achieved by a Xerox 820 operating under CP/M and programmed in RMCobol. This combination took 5740 seconds to complete the same task — a difference of approximately 700,000 times in the speed of execution. Remember that the performance of a machine on a single program may say very little about the utility of the machine and language combination for your particular purposes.

However, that said, the fascinating statistic that I found in the listing of performance figures was for the Apple II computer programmed in Fortran. One version of Fortran running on the original 6502 central processor unit took 333 seconds to find all the expected prime numbers.

A second version of Fortran which operated under CP/M using a plug-in Microsoft card took only 34 seconds, about one tenth of the time. That difference cannot be explained by variation in the clock rates allowing one processor to run faster than another, nor can the physical construction of the machine play any significant part. The only possible explanation is that the version of Fortran running under CP/M is vastly more efficient than the other for this purpose — a better implementation of the language. ∎

Do you have a problem? Your manual is incomprehensible or you just cannot get the hang of that programming trick you tried — whatever it is, Tim Hartnell will do his best to answer your queries. Please include only one question per letter and mark them "Response Frame".

ACE MEMORY

■ I have recently bought a Jupiter Ace. I find the 3K on board rather restrictive. What is the easiest way to expand the memory, and what size can I go up to?

Donald Towlinson,
Yate,
Bristol.

A 16K RAMPACK for the Ace is available from Stonechip Electronics, Unit 9, The Brook Industrial Estate, Deadbrook Lane, Aldershot, Hampshire. The company told me that although it is supplied as a 16K unit, it can be expanded — using chips available from them — to 32K. The extra 16K costs £19.95. You can get more details by calling the company on 0252-318260. If you want to discuss your computer with other Ace owners, you might like to get in touch with the users' club. Send a stamped, addressed envelope to Remsoft, 18 George Street, Brighton, BN2 1RH.

BOARD GAMES

■ My particular computer interest is in playing board games but I would like to enlarge the normal miniscule playing board from eight by eight to 24 by 24. I gather that this can easily be done on the ZX-81 by Poking into the variable which controls the bottom half of the screen. If, however, I Poke 23659 with zero on the Spectrum I get a blank screen, remedied by altering the Border colour, but the computer then locks up. Is there a solution?

D G Daverson,
Brentwood.

THERE ARE two ways around this. Either use Print hash, or change your board size to 16 by 16. This allows you to use each square on the board occupying four character cells, and some pretty elaborate pieces can be created with user-defined graphics to fit within the 16 by 16 matrix.

MISSILE GAP

■ I own an Atari 400 computer. Knowing that you have one, I decided to enquire about some things which I do not understand. I have been told that there are player-missile graphics but none of the numerous books and papers which I got with the computer signified what player-missile graphics are, or how they can be operated. Can you enlighten me? I would also like to know how one can use the user-definable graphics characters.

Thomas Martin,
Kerridge,
Macclesfield.

PAUL BUNN, in his book *Making the Most of Your Atari*, explains that there are four players and four missiles on the computer, each with its own colour register, size register and horizontal position register. A player is exactly eight bits wide by either 128 — double — or 256 — single-line resolution. To create your player, you plot the squares on a grid which is eight by whatever height you want the player to be. You then put the numbers 128, 64, 32, 16, 8, 4, 2 and 1 above the grid, adding the value if the square for the shape is filled in.

ORIC NOTICE

■ I am thinking of buying an Oric computer. I will be using it, among other things, as a kind of electronic bulletin board for a club I belong to. I have seen that you can get tall letters on the BBC Micro. Is it possible to do the same with the Oric? I cannot afford a BBC machine, and have heard that the Oric would be a good second choice.

Patrick Shepson,
Leighton Buzzard.

THE ORIC has a number of commands — including the colour ones, of course — to enable you to dress up your text output. You can get double-height characters by using Control D. If you want double-height characters, you enter as part of the program
PRINT CHR$(4);CHR$(27);"J
....followed by text".
The J will not be printed. Change the J to an N to get double-height, flashing characters.

WORD PROCESS

■ I am a newcomer to micro-computers and with *Your Computer* and a subscription to some books, have been trying to learn more about the mountain of hardware and software available. I want to be able to do word processing but it seems that as printers go from 80 characters upwards, to have a micro-computer displaying 32 characters is not the best way of doing it. To have to scroll a 32-character screen left to right does not seem ideal. As I am an accountant, I want to be able to compile financial software. But I feel that disc drives will make some computers I am considering, such as the NewBrain, too expensive. The Spectrum for the price seems excellent, especially with the Microdrive. Could you give me an assessment of the home-study Basic course offered by the NCC. Is it of any use to a beginner?

B J Owen,
Riyadh,
Saudi Arabia.

A BBC MICRO with word processor ROM would solve your word processing problems, but you would have to check that a suitable accountancy package is available; and also whether you can afford to add discs to the BBC machine when you get it. The NCC course has sold extremely well. While not the breeziest presentation of the subject matter available, it covers the field very competently and carefully. It is one of the best overall introductions to Basic available at the moment.

INCOMPATIBILITY

■ I am still waiting for my Spectrum to be delivered. I do not know much about the machine code. I would like to know if I can use the ZX-81 machine-code articles featured in the past in *Your Computer*, and if not, what modifications would be needed to get them to work.

Farid Ehsan,
Portswood,
Southampton.

UNFORTUNATELY, the short answer is "No". Most machine-code programs put material on to the screen, and the routines to do this need to be totally rewritten for the Spectrum.

DRAGON DAZE

■ I have seen a great deal of software advertised for the Dragon 32. I have only had my Dragon for a month, and am quite bewildered as to which is the best software to buy. I am mainly interested in arcade games. Can you advise me?

Gwyn Parkly,
Paisley,
Renfrewshire.

THE SCOTTISH Dragon Club, 1 Walker Street, Edinburgh may well be able to help you with information on commercial software. Suppliers of Dragon software include APG. Software, 24 Mountain View, Peel, Isle of Man; Stanley Software, 91 Cradley, Widnes, Cheshire; Saint George Software, 73 Ling Street, Liverpool, L7; DACC, 23 Waverley Road, Hindley, Greater Manchester, WN2 3BN; Juniper Computing, 8 Pembroke Green, Lea, Malmesbury, Wiltshire SN16 9PB; and Wizard Software, PO Box 23, Dunfermline, Fife, KY11 5RW. There are also a number of Dragon books around including *Dragon Load and Go*, *Making the Most of Your Dragon 32*, *Enter the Dragon* and *Dynamic Games for the Dragon 32*. See also last month's review of Dragon software in *Your Computer*.

DRAGON LORE

■ I have recently bought a Dragon 32 computer and would like to know if you could recommend a book on programming in extended Basic. Also, could you tell me which other computer software is compatible with the Dragon?

Mrs J Begg,
Abergavenny,
Gwent.

THERE ARE two books on programming in Tandy Color Computer Basic available from Tandy dealers. Although these books deal with the Tandy Color Computer, they apply to the Dragon as the two computers appear to have virtually identical ROMs. Because of this, Tandy Color Computer programs will run on the Dragon 32, if you manage to buy the software on cassette. As well as the books sold by Tandy, there are several other books on programming the Tandy Color Computer available in the U.K. which should prove of use. You will probably find them advertised in this issue of *Your Computer*.

ROM SWEET ROM

■ Reading the BBC Micro-computer literature, I see that "up to four different 16K interpreters" may be fitted inside the machine, and that the Basic interpreter supplied is classed as one of these. Are the other language ROMs — Pascal, Forth and Lisp — available yet? Is the mnemonic assembler part of the Basic ROM or is it part of the 16K machine's operating system, and therefore able to be accessed by the other language ROMs? Will it be possible to buy the Micro fitted with one of the other languages as standard instead of the Basic interpreter being supplied?

M E Needham,
Clitheroe,
Lancashire.

THE OTHER language ROMs are being developed in order, with the Pascal one first. The assembler is part of the Basic ROM, and therefore cannot be accessed by the other language ROMs. However, it is possible that the other ROMs may include assemblers. Acorn tells me it has no plans to release the computer with any onboard language, as the initial choice, except Basic. ■

SOFTWARE FILE

Software File gives you the opportunity to have your programs, ideas and discoveries published. We will accept contributions for any home computer provided they are submitted to *Your Computer* exclusively. Please double-check your programs and specify the memory they require before sending them, preferably on cassette. We pay between £6 and £36 for contributions published.

Millipede

Shingo Sugiura,
Putney,
London.

BBC

YOU MUST guide a hungry baby millipede gobbling up red fruits whilst avoiding the deadly rocks. Naturally, if you guide him

backwards he will bite himself and die. Every time he eats a fruit he grows by one segment. When it reaches a length of 40 segments, you will have to guide another faster baby millipede on a new screen.

You have 3 lives at your disposal. The score is constantly updated and because all the calculations and the printing of the millipede's segments are done in machine code, the action is very fast. If you want to change your speed

press Escape, this will revert you back to the instructions without clearing the high score. Use Z & X for left and right respectively, and the colon and oblique stroke for up and down.

Lines		
Lines	50 to 240	machine code
Lines	250 to 330	sets up screen
Lines	340 to 500	main program
Lines	520 to 700	initialise variables
Lines	710 to 900	instructions
Lines	910 to 1030	sets up screen
Lines	1040 to 1090	prints the rocks
Lines	1110 to 1200	prints the fruits
Lines	1210 to 1240	prints score and high score
Lines	1250 to 1280	new screen procedure
Lines	1290 to 1310	checks the colour in front of head
Lines	1320 to 1360	death procedure
Lines	1370 to 1390	ending procedure
Lines	1400 to 1420	delay procedure

```
  2 REM ****** MILLEPEDE ******
  3 REM ** (c) March 1983 **
  4 REM ** By Shingo Sugiura **
  5 REM *********************
  6
 10 ON ERROR GOTO 30
 20 hi%=50
 30 MODE7:PROCinst
 40 HIMEM=&2800:OSWRCH=&FFEE:LGTH%=&70:HEAD%=&71
 50 FOR I%=0 TO 2 STEP 2:P%=&2800
 60   [OPTI%
 70   LDY #40
 80   .LOOP
 90   LDA &2C1F,Y
100   STA &2C20,Y
110   LDA &2C51,Y
120   STA &2C52,Y
130   DEY
140   BNE LOOP
150   LDY (LGTH%)
160   .ANI
170   LDA #31:JSR OSWRCH
180   LDA &2C20,Y:JSR OSWRCH:LDA &2C52,Y:JSR OSWRCH
190   LDA #225:JSR OSWRCH
200   DEY
210   BNE ANI
220   LDA #31:JSR OSWRCH:LDA &2C20:JSR OSWRCH:LDA &2C52:JSR OSWRCH
230   LDA HEAD%:JSR OSWRCH
240   RTS:]NEXT I%
241
250 score%=0:life%=3:screen%=1
260 HIMEM=&2800
270 VDU22,1,23,10,32,0,0,0,0,19,3,2,0,17,1,17,131,12
280 HIMEM=&2800
290 PROCdata
300 PROCinit
310 PROCwall
320 PROCobst
330 PROCfood
340 REPEAT
350 IF INKEY(-98)=-1 THEN A$="LEFT"
360 IF INKEY(-67)=-1 THEN A$="RIGHT"
370 IF INKEY(-73)=-1 THEN A$="UP"
380 IF INKEY(-105)=-1 THEN A$="DOWN"
390 PRINTTAB(?(&2C20+?LGTH%),?(&2C52+?LGTH%))," "
400 IF A$="LEFT" THEN ?&2C20=(?&2C20)-1:?HEAD%=229
410 IF A$="RIGHT" THEN ?&2C20=(?&2C20)+1:?HEAD%=228
420 IF A$="UP" THEN ?&2C52=(?&2C52)-1:?HEAD%=226
430 IF A$="DOWN" THEN ?&2C52=(?&2C52)+1:?HEAD%=227
440 VDU26:col%=POINT(?&2C20*32+16,(31-?&2C52)*32+16)
450 IF col%<>3 THEN PROCcheck
460 PROCnumerics
470 CALL &2800
480 PROCwait(speed%)
490 UNTIL ?LGTH%=40:PROCnew_screen
500 GOTO270
510             *******************************
520 DEFPROCinit
530 ENVELOPE1,1,3,2,-2,6,6,6,100,0,0,-5,100,0
540 VDU23,224,255,255,255,255,255,255,255,255
550 VDU23,225,24,36,126,189,189,126,36,24
560 VDU23,226,66,36,24,126,219,126,60,24
570 VDU23,227,24,60,126,219,126,24,36,66
580 VDU23,228,16,57,106,252,252,106,57,16
590 VDU23,229,8,156,86,63,63,86,156,8
600 VDU23,230,0,24,40,92,126,190,127,120
610 col%=3
620 A$="RIGHT":?LGTH%=2
630 Screen=1
640 ENDPROC
650 DEFPROCdata
660 FOR Xdata=0 TO 40
670 ?(&2C20+Xdata)=Xdata+15:NEXT Xdata
680 FOR Ydata=0 TO 40
690 ?(&2C52+Ydata)=15:NEXT Ydata
700 ENDPROC
701
710 DEFPROCinst
720 PRINTTAB(13,1);CHR$(130);CHR$(141)"MILLEPEDE"
730 PRINTTAB(13,2);CHR$(130);CHR$(141)"MILLEPEDE"
740 PRINT CHR$(131);TAB(11)"BY SHINGO SUGIURA"
750 PRINT'CHR$(132);"You must guide the growing millipede"
760 PRINT CHR$(132);"towards the ripe fruits while avoiding"
770 PRINT CHR$(132);"the rocks ."
780 PRINT'CHR$(132);"If you try to guide him backwards he"
790 PRINT CHR$(132);"will bite himself and die."
800 PRINT CHR$(132);"  You have 3 lives."
810 PRINT' You to change level press ESCAPE .."
820 PRINT'CHR$(133),"CONTROLS :-"
830 PRINT'CHR$(134);"UP ----------- '/'"
840 PRINT'CHR$(134);"DOWN ----------- '/'"
850 PRINT'CHR$(134);"LEFT ----------- 'Z'"
860 PRINT'CHR$(134);"RIGHT ----------- 'X'"
870 PRINT'" Input speed (0 Fast to 9 slow) ";:REPEATI%=INSTR("0123456789",GET$)
    :UNTILI%:PRINT;I%-1:speed%=(I%-1)*10
880 PRINT'CHR$(136);CHR$(130);TAB(7);"Press SPACE BAR to start";
890 REPEAT UNTIL GET=32
900 ENDPROC
901
910 DEFPROCwall:COLOUR0
920 VDU31,13,0,226:PRINT" MILLEPEDE ";CHR$226
930 PRINT TAB(2,3);STRING$(36,CHR$224)
940 PRINT TAB(2,28);STRING$(36,CHR$224)
950 FOR WALL=2 TO 27
960   PRINTTAB(2,WALL)CHR$224
970   PRINTTAB(37,WALL)CHR$224
980   SOUND &11,-10,WALL*2+100,2
990   PROCwait(50)
1000  NEXT WALL
1010 PRINT TAB(3,30);"Score ";score%;TAB(20,30);"High score ";hi%
1020 PRINT TAB(3,31);"Lives left ";life%;TAB(20,31);"Screen ";screen%
1030 ENDPROC
1031
1040 DEFPROCobst
1050 FOR OBST=1 TO screen%*3+10
1060   PRINTTAB(RND(33)+3,RND(24)+2)CHR$230
1070   SOUND0,-15,6,1
1080   PROCwait(100)
1090   NEXT OBST:ENDPROC
1100
1110 DEFPROCfood:COLOUR1
1120 FOR FOOD=1 TO 50
1130   PROCwait(100)
1140   SOUND1,-10,RND(200),1
1150   PRINTTAB(RND(33)+3,RND(24)+2)"*"
1160   NEXT FOOD
1170 VDU17,0,31,17,15,228,17,1
1180 PRINT TAB(18,15)" Ready !":PROCwait(2000)
1190 PRINT TAB(18,15);" "
1200 ENDPROC
1201
1210 DEFPROCnumerics:COLOUR0
1220 IF score%>hi% THEN hi%=score%
1230 PRINT TAB(9,30);score%;TAB(31,30);hi%
1240 ENDPROC
1241
1250 DEFPROCnew_screen
1260 PROCwait(2000)
1270 screen%=screen%+1:speed%=speed%-10:IF speed%<0 THEN speed%=0:CLG
1280 ENDPROC
1281
1290 DEFPROCcheck
1300 IF col%=1 THEN score%=score%+1:?LGTH%=(?LGTH%)+1:SOUND1,1,1,3:ENDPROC
1310 IF col%=0 THEN PROCdead:ENDPROC
1311
1320 DEFPROCdead
1330 SOUND0,1,50,5
1340 PROCwait(1500)
1350 life%=life%-1:IF life%<0 THEN PROCend
1360 CLG:GOTO270:ENDPROC
1361
1370 DEFPROCend
1380 PRINT"DEAD":PRINT"Press SPACE BAR to replay ":REPEAT UNTIL GET=32
1390 GOTO250:ENDPROC
1391
1400 DEFPROCwait(T%)
1410 FOR Delay%=0 TO T%:NEXT Delay%
1420 ENDPROC
```

Surrounded
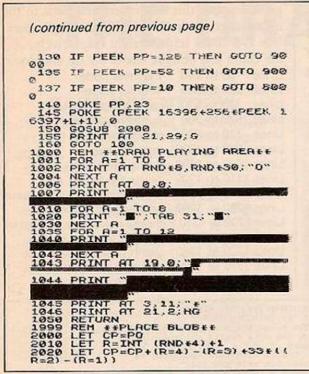
Mark Easton,
Mapperley,
Nottingham.

ZX-81

SURROUND IS a novel addictive game for the ZX-81, requiring a good strategic mind. The program takes up just under 4K of memory. The instruction and a sample of the graphics

are shown with the listing. In order to make the program easier to understand, Rem statements are included. The program gives a good example of how to Peek and Poke into the display file. Another interesting point is line 2020. This one line replaces:

```
IF R = 4 THEN LET CP = CP + 1
IF R = 3 THEN LET CP = CP - 1
IF R = 2 THEN LET CP = CP + 33
IF R = 1 THEN LET CP = CP + 33
```

```
  1 REM
  2 REM
  3 LET HG=0
  5 GOSUB 9100
 10 LET PO=110
 20 GOSUB 1000
 30 LET Z=6
 40 LET G=0
100 LET I$=INKEY$
105 IF I$="" THEN GOTO 100
110 LET L=PO
115 IF (CODE I$)>36 OR (CODE I$
)<33 THEN GOTO 100
120 LET PO=PO+(I$="8")-(I$="5")
+33*((I$="6")-(I$="7"))
125 LET G=G+1
127 LET PP=PEEK 16396+255*PEEK
16397+PO+1
```

(continued on next page)

(continued from previous page)

```
 130 IF PEEK PP=128 THEN GOTO 90
00
 135 IF PEEK PP=52 THEN GOTO 900
0
 137 IF PEEK PP=10 THEN GOTO 800
 140 POKE PP,23
 145 POKE (PEEK 16396+256*PEEK 1
6397+L+1),A
 150 GOSUB 2000
 155 PRINT AT 21,29;G
 160 GOTO 100
1000 REM **DRAW PLAYING AREA**
1001 FOR A=1 TO 6
1002 PRINT AT RND*8,RND*30; "O"
1004 NEXT A
1006 PRINT AT 0,0;
1007 PRINT "
1010 FOR A=1 TO 8
1020 PRINT "  "; TAB 31; "
1030 NEXT A
1035 FOR A=1 TO 12
1040 PRINT "
1042 NEXT A
1043 PRINT AT 19,0;"
1044 PRINT "
1045 PRINT AT 3,11;"*"
1046 PRINT AT 21,2;G
1047 PRINT AT 21,29;HG
1050 RETURN
1999 REM **PLACE BLOB**
2000 LET CP=PO
2010 LET R=INT (RND*4)+1
2020 LET R=CP+(R=4)-(R=3)+33*((
R=2)-(R=1))
```

```
2025 LET PZ=PEEK 16396+256*PEEK
16397+CP+1
2030 IF PEEK PZ=128 THEN GOTO 30
00
2035 IF PEEK PZ=52 THEN GOTO 210
0
2040 POKE PZ,128
2050 RETURN
2099 REM **EXPAND FRAME**
2100 LET Z=Z+1
2102 IF Z>18 THEN LET Z=18
2105 POKE PZ,0
2107 POKE PP,23
2110 PRINT AT Z,1;"
2115 PRINT AT TAB 1;"
2120 PRINT AT Z,(RND*28)+1; "O";A
T Z+1,(RND*28)+1; "O"
2130 RETURN
7999 REM **YOU WIN**
8000 PRINT AT 0,8;"WELL DONE"
8002 POKE (PEEK 16396+256*PEEK 1
6397+L+1),0
8004 POKE PP,151
8010 GOTO 9030
8999 REM **YOU ARE DESTROYED**
9000 POKE (PEEK 16396+256*PEEK 1
6397+L+1),0
9010 POKE PP,8
9020 PRINT AT 0,10;"DESTROYED"
9030 IF G<=HG THEN GOTO 9050
9040 PRINT AT 1,3;"YOU HAVE [MR
9045 LET HG=G
9046 PRINT AT 20,8;"
9050 PRINT AT 20,8;"HIT ANY KEY"
9060 PRINT AT 21,29;G
9065 IF INKEY$<>"" THEN GOTO 906
9070 IF INKEY$="" THEN GOTO 9070
9075 CLS
9080 GOTO 10
```

```
9100 REM **INSTRUCTIONS**
9110 PRINT TAB 11; "SUROUND"
9120 PRINT
9130 PRINT "   THE GAME TAKES PL
ACE IN A", "BLACK FRAME CONTAININ
G ABOUT 6", "DEATH PITS 'O', WHI
CH YOU MUST", "AVOID. YOU PEICE I
S **"
9140 PRINT "  HOWEVER, AFTER YO
UR TURN, THE", "COMPUTER WILL PLAC
E A BLOB", "■", "IN A VACANT SQU
ARE IN AN ATTEMPT TO SUROUND YOU.
  IF YOU HIT A", "BLOB THEN YOU AR
E EVAPOURATED."
9150 PRINT "   BUT IF YOU MANAGE
 TO MAKE A", "BLOB LAND ON A PIT
 YOU WILL SEE", "THAT THE FRAME EX
PANDS. THE ONLY WAY TO SURVIVE CE
RTAIN DEATH IS", "TO REACH THE GR
AY LINE, WHICH", "CAN ONLY BE REA
CHED BY FULLY", "EXPANDING THE FR
AME. BUT DON'T", "EXPAND IT PAS
T THE GRAY LINE."
9160 PRINT AT 21,0; "HIT ANY KEY
 TO CONTINUE"
9170 IF INKEY$="" THEN GOTO 9170
9180 CLS
9190 GOSUB 1000
9200 PRINT AT 10,1; "           THIS IS T
           USE
   THE ARROWED KEYS
   THEN PRESS 'P' UNTIL
   THERE IS NO NEED
9210 PRINT AT 21,0; "HIT ANY KEY
9220 IF INKEY$<>"" THEN GOTO 922
9230 IF INKEY$="" THEN GOTO 9230
9240 CLS
9250 RETURN
9998 SAVE "SUROUND"
9999 RUN
```

Block blitz

*Robert Newton,
Wadhurst,
Sussex.*

SPECTRUM

THIS COMBINES attractive graphics with very interesting sound effects. It is based upon the Vic-20 game of Blitz and helpfully tells you how many people you massacred if your aircraft runs into too many tower blocks. A high-score routine and variable number of chances have been included to add the competitive nature of the game. If the subroutine at line 7000 is included, then it will start up with some attractive large characters using the machine-code routine of the Horizon cassette which every Spectrum owner will have. Just run one of the programs on the Horizons cassette and then break the program and type in as a direct command:

SAVE "mc" CODE 32256, 300

to be saved, after the Blitz program on cassette.

```
10 REM       blitz
11 RESTORE 4: LET hc=1000: LET
k=0: BORDER 4: PAPER 0: INK 7: C
LS
20 REM set up graphics
30 DATA 255,60,126,255,255,255
,18,24
110 FOR n=0 TO 7
120 READ a
130 POKE USR "a"+n,a
140 NEXT n
150 REM bomb= "A"
160 DATA 24,24,102,102,219,219,
219,255
240 FOR n=0 TO 7
250 READ a: POKE USR "b"+n,a
260 NEXT n
270 REM "C"
280 DATA 219,219,219,255,219,21
9,219,219
350 FOR n=0 TO 7
370 READ a
380 POKE USR "c"+n,a
390 NEXT n
400 REM "C"
410 DATA 231,231,231,255,255,23
1,231,231
480 FOR n=0 TO 7
500 READ a
510 POKE USR "d"+n,a
520 NEXT n
530 REM "C"
540 DATA 24,24,24,24,60,126,231
,231
550 FOR n=0 TO 7
600 READ a
640 POKE USR "e"+n,a: NEXT n
650 REM "C"
660 DATA 24,60,126,219,153,153,
255,255
740 FOR n=0 TO 7
750 READ a
760 POKE USR "f"+n,a
770 NEXT n
780 DATA 255,153,255,153,255,15
3,255,255
860 FOR n=0 TO 7
880 READ a: POKE USR "g"+n,a
890 NEXT n
900 DATA 255,146,73,146,73,146,
73,255
920 DATA 146
930 DATA 73
940 DATA 146
950 DATA 73
960 DATA 146
970 DATA 73
980 DATA 255
990 FOR n=0 TO 7
1000 READ a: POKE USR "h"+n,a
1010 NEXT n
1020 REM "H"
1030 CLS
1035 INK 2: BRIGHT 0
1040 PRINT AT 21,0; "HHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHH"
1050 FOR y=20 TO 14 STEP -1
1060 PRINT AT y,2; "CC"; AT y,5; "
CC"; AT y,9; "DG"; AT y,15; "C"; AT
y,20; "CGG"; AT y,24; "G"; AT y,2
6; "CDDGC"
1090 NEXT y
1100 PRINT AT 13,2; "CF"; AT 13,5;
"F"; AT 13,7; "G"; AT 13,9; "EG"; AT
13,12; "BC"; AT 13,15; "FGE"; AT 13,
19; "BCGG"; AT 13,24; "G"; AT 13,26;
"CEDGB"
1110 PRINT AT 12,2; "C"; AT 12,5; "
F"; AT 12,7; "G"; AT 12,10; "G"; AT 1
```
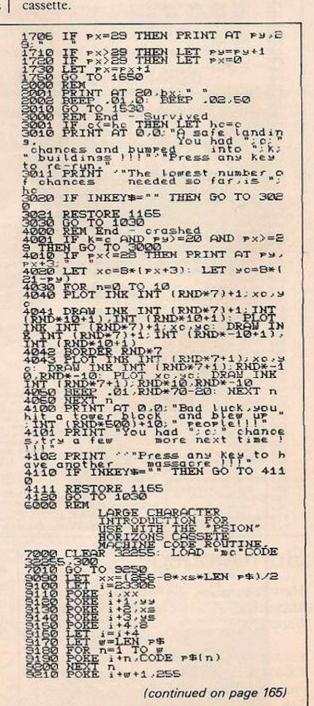
```
2,13; "C"; AT 12,16; "B"; AT 12,20; "
EGF"; AT 12,26; "C"; AT 12,28; "BE
G"
1120 PRINT AT 11,2; "C"; AT 11,7; "
G"; AT 11,10; "B"; AT 11,13; "C"; AT
11,21; "B"; AT 11,26; "C"
1130 PRINT AT 10,2; "B"; AT 10,7; "
C"; AT 10,13; "C"; AT 10,26; "B"
1140 PRINT AT 8,7; "F"; AT 8,13; "B
1151 INK 7: BRIGHT 0
1165 DATA 0,112,248,248,255,255,
255,15
1240 RESTORE 1165: FOR n=0 TO 7
1250 READ a: POKE USR "i"+n,a
1260 NEXT n
1270 DATA 7,1,50,116,255,255,255
1350 FOR n=0 TO 7
1360 READ a: POKE USR "j"+n,a
1370 NEXT n
1380 DATA 248
1390 DATA 32
1400 DATA 64
1410 DATA 128
1420 DATA 248
1430 DATA 240
1440 DATA 248
1450 DATA BIN 0
1470 FOR n=0 TO 7
1480 READ a: POKE USR "k"+n,a
1490 NEXT n
1500 LET py=0: LET k=0
1510 LET by=0: LET bx=0
1520 REM     start game
1525 INPUT "How many chances ?";
c
1530 IF px>0 THEN PRINT AT py,px
1540 PRINT AT py,px; "IJK"
1541 IF ATTR (py,px+3)=2 THEN LE
T k=k+1
1542 BEEP .001,10: BEEP .002,30:
BEEP .001,55
1543 IF k>=c THEN GO TO 4000
1544 IF px>=29 THEN LET py=py+1
1545 IF px>=29 AND py=21 THEN G
O TO 3000
1575 IF px=29 THEN PRINT AT py,
1,"
1580 IF px>=29 THEN LET px=0
1590 LET px=px+1
1595 LET a$=INKEY$
1600 IF INKEY$="" THEN GO TO 153
1610 REM do bombs
1611 BEEP .004,0: BEEP .003,2: BE
EP .002,4: BEEP .002,6: BEEP .002,8
02,10: BEEP .002,12: BEEP .002,2
1620 IF py>=20 THEN LET by=21
1630 IF px<31 THEN LET bx=px+1
1640 IF px=32 THEN LET bx=32
1645 IF px<=29 THEN LET bx=32
1660 IF by=21 THEN PRINT AT 21,b
x; "     "
1665 BEEP .001,65
1670 IF by=21 THEN GO TO 2000
1680 IF by>1 THEN PRINT AT by-1,
bx; " "
1681 BEEP .001,60-by
1690 LET by=by+1
1700 IF by=21 AND px>=29 THEN GO
TO 3000
1701 PRINT AT py,px-1; " "
1702 PRINT AT py,px; "IJK"
1703 IF px>28 THEN PRINT AT py,2
9; "
1704 IF ATTR (py,px+4)=2 THEN LE
T k=k+1
1705 IF k>=c THEN GO TO 4000
```

```
1706 IF px=29 THEN PRINT AT py,2
9;
1710 IF px>29 THEN LET py=py+1
1720 IF px>29 THEN LET px=0
1730 LET px=px+1
1750 GO TO 1650
2000 REM
2001 PRINT AT 20,bx; "    "
2005 BEEP .01,0: BEEP .02,50
2010 GO TO 1530
3000 REM End - Survived
3001 IF c<=hc THEN LET hc=c
3010 PRINT AT 0,0; "A safe landin
g.           You had    c
chances and bumped    into    k
buildings !!!","Press any key
to re-run."
3011 PRINT "The lowest number of
   chances   needed so far is ";
c
3020 IF INKEY$="" THEN GO TO 302
0
3021 RESTORE 1165
3030 GO TO 1030
4000 REM End - crashed
4001 IF k=c AND py>=20 AND px=2
9 THEN GO TO 3000
4010 IF px<=28 THEN PRINT AT py,
px+3; "
4020 LET xc=8*(px+3): LET yc=8*(
21-py)
4030 FOR xo=8 TO 10
4040 PLOT INK INT (RND*7)+1; xc,y
c
4041 DRAW INK INT (RND*10+1); INT
(RND*10+1): PLOT
INK INT (RND*7)+1; xc,yc: DRAW IN
K INT (RND*7)+1; INT (RND*-10+1);
INT (RND*10+1)
4042 BORDER RND*7
4043 PLOT INK INT (RND*7)+1; xc,y
c: DRAW INK INT (RND*7)+1; RND*-1
0,RND*-10: PLOT xc,yc; DRAW INK
INT (RND*7+1); RND*10,RND*10
4050 BEEP .01,RND*70-20: NEXT n
4060 NEXT n
4100 PRINT AT 0,0; "Bad luck,you
hit a tower block   and blew up !
  INT (RND*500)+10; people !!"
4101 PRINT "  You had "; c; " chance
s,try a few     more next time !"
4102 PRINT "  Press any key to h
ave another    massacre !"
4110 IF INKEY$="" THEN GO TO 411
0
4111 RESTORE 1165
4120 GO TO 1030
6000 REM
         LARGE CHARACTER
         INTRODUCTION FOR
         USE WITH THE "PSION"
         HORIZONS CASSETTE
         MACHINE CODE ROUTINE
7000 CLEAR 32255: LOAD "mc" CODE
32256, 300
7010 GO TO 9250
9090 LET xs=(255-8*xs*LEN p$)/2
9100 LET i=23306
9110 POKE i,xx
9120 POKE i+3,xs
9130 POKE i+4,ys
9140 POKE i+5,xs
9150 LET i=i+4
9170 LET w=LEN p$
9180 FOR n=1 TO w
9190 POKE i+n,CODE p$(n)
9200 NEXT n
9210 POKE i+w+1,255
```

SOFTWARE FILE

```
3220 LET w=USR 32256
3230 RETURN
3250 OVER 0: PAPER 7: BORDER 2:
FLASH 0: INVERSE 0: BRIGHT 1: IN
K 1: CLS
3260 LET p$="© Robert Newton": L
ET ys=1
3270 LET xs=2: LET ys=3
3280 GO SUB 9090
3290 LET ys=60: LET p$="BLITZ": LET
FLASH 0: INK 3: LET xs=6: LET ys
=7: GO SUB 9090
9300 FLASH 0: INK 0: LET p$="Pre
ss any key to continue": LET xs=
1: LET ys=3: LET yy=166: GO SUB
9090
9310 PAUSE 0
9320 RUN
```

Juggler

J Charlesworth,
Kings-Lynn,
Norfolk.

SPECTRUM

THE OBJECT of this game, for the 16 or 48K ZX Spectrum, is to keep the balls which are bouncing around the screen from hitting the bottom of the screen. To do this, you move a bat back and forth across the bottom of the screen, trying to make the balls land on it so they can rebound upwards.

At first there is just one ball, but every 25 seconds another ball is added and your points, displayed in the top right-hand corner, go up in steps of however many balls there are. All the controls are printed by the computer on running the program.

For speed, I have written all the moving graphics section in machine code. This is stored in the data statements from line 9000 onwards, which the computer will check for errors with a checksum when it is run.

```
  1 BORDER 0: PAPER 0: INK 7
 10 GO TO 9000
 20 LET a$="           JUGGLER
": LET x=10: GO SUB
200
 30 LET a$="    5 (-
->8 ": LET x=12: GO SUB
200
 40 LET a$="   H=HOLD
S=START ": LET x=14: GO SUB
200
200 FOR a=1 TO 300
200 FOR a=1 TO LEN a$: PRINT AT
x,0; a$(LEN a$-a+1 TO
); BRIGHT 1;a$(LEN a$-a+1 TO
): BEEP .01,a+x: NEXT a: RETURN
300 FOR a=1 TO 10: POKE 32346+4
*a,INT (RND*6)+3: POKE 32347+4*a
,INT (RND*8)+2: POKE 32348+4*a,
1-(2 AND RND>.5): POKE 32349+4*a
,1-(2 AND RND>.5): NEXT a
310 LET a$="          PRESS 5
": LET x=21: GO SUB
200
320 GO SUB 500
300 CLS : PRINT AT 0,0;: LET t=
USR 32000
340 LET r=0: LET p=0: GO SUB 40
0
350 IF INKEY$="h" THEN GO SUB 5
00
360 IF PEEK 23673>=5 THEN GO SU
B 300
380 LET p=p+r: PRINT AT 0,27; p
APER 1,p; PAPER 0: LET t=USR 325
27: IF PEEK 23296<>99 THEN GO TO
350
390 GO TO 300
400 POKE 23672,0: POKE 23673,0:
LET r=r+(1 AND r<>0): POKE 325
31,r: POKE 32575,r: POKE 32578,3
0-2*r: PRINT AT 0,1; PAPER 1;"RO
```

```
UND ";r;AT 0,20;"POINTS ";p: RET
URN
500 LET t=PEEK 23672: LET t1=PE
EK 23673
510 IF INKEY$<>"s" THEN GO TO 5
10
520 POKE 23672,t: POKE 23673,t1
: RETURN
9000 DATA 62,17,215,62,1,215,62,
6,215,62,5,215,6,21,62,22,215,12
0,215,62,0,215,62,16,215,62,62,2
15,120,215,62,31,215,62,32,215,1
6,-24,33,175,90,34,176,92,6,3,54
,87,35,16,-5
9010 DATA 201
9020 DATA 62,0,50,129,92,1,254,2
2,237,120,203,87,62,1,14,16,176,
2,24,186,140,7,62,1,50,129,92,44
,24,1,254,247,237,120,203,103,32
,14,50,176,92,24,254,161,40,7,62,25
5,50,129,92,24,1,201
9030 DATA 58,129,92,254,1,32,16,
42,176,92,54,7,35,35,35,54,87,43,
43,34,176,92,24,14,176,92,24,5
4,87,34,176,92,35,35,54,7,201
9050 DATA 62,22,215,42,0,91,125,
215,124,215,62,16,215,62,7,215,6
2,32,215,237,91,2,91,125,62,7,215,6
124,130,95,205,244,126,254,7,32,
23,33,0,91,112,35,115,62,22,215,
120
9060 DATA 215,215,215,62,16,215,
52,4,215,62,144,215,201,58,0,91,
58,2,91,134,71,35,94,205,244,126,
254,7,32,10,58,3,91,237,68,50,3
,91,24,39,33,0,91,70,35,58,3,91,
134,95,205,244,126,254,7
9070 DATA 58,10,58,2,91,237,68,5
0,2,91,24,12,33,3,91,126,237,68,
119,35,126,237,68,119,33,0,91,58
```

```
,2,91,134,119,35,58,3,91,134,119
,43,70,35,94,62,22,215,120,215,1
35,215,62,16
9080 DATA 215,62,4,215,62,144,21
5,33,0,91,120,0,205,181,3,201
9081 DATA 193,225,6,10,197,33,50
,0,6,0,197,229,17,4,0,205,181,3,
225,193,35,16,-13,193,16,-22,62,
99,50,0,91,201
9090 CLEAR 31999: LET a$="
PLEASE WAIT ": LET
x=10: GO SUB 200
9095 LET t=0: FOR a=32000 TO 323
42: READ b: LET t=t+b: POKE a,b:
NEXT a: IF t<>31720 THEN GO TO
9999
9100 RESTORE 9105: LET t=0: FOR
a=32500 TO 32593: READ b: LET t=
t+b: POKE a,b: NEXT a: IF t<>105
58 THEN GO TO 9999
9105 DATA 197,213,229,213,62,0,1
84,40,9,17,32,0,35,0,68,25,16,-3
,209,22,0,25,126,225,209,193,201
9110 DATA 33,94,126,6,1,197,229,
170,91,237,160,237,160,237,160,
237,160
9120 DATA 205,142,125,225,17,0,9
1,235,126,254,21,202,65,126,237,
160,237,160,237,160,237,160,235,
193,16,-39
9130 DATA 205,52,125,6,2,197,6,3
0,197,6,0,16,-2,193,16,-8,205,52
125,193,16,-17,201
6140 RESTORE 9140: FOR a=0 TO 7:
READ b: POKE USR "a"+a,b: NEXT
a: DATA 60,126,255,255,255,255,1
26,60
9150 GO TO 20
9999 PRINT AT 10,7; FLASH 1;"
DATA ERROR       ": STOP
```

Sound analysis

Predrag Bogdanovic,
Belgrade,
Yugoslavia.

ZX-81

THIS MACHINE-CODE program for the ZX-81 draws graphics illustrating sound. Sound is fed in on the Ear socket. To Load program create a Rem line 45 characters long. Use any hexloader to enter the code. The program is executed by RAND USR 16526. If you want the computer to draw one graphic and then return to Basic type:

```
POKE 16551,192(N/L)
```

Sub hunt

Jason Bumford,
Hanham,
Bristol,

VIC-20

SUB HUNT runs in an unexpanded Vic-20. It shows a minesweeper on the sea's surface, which moves left and right under machine-code control, while the rest of the Basic program controls submarines passing at random depths and directions. The minesweeper can drop depth charges from the bow and stern and the submarines will randomly retaliate with torpedos. Each sub sunk scores 10 points, and a deliberately wasted depth charge losses two points. A torpedo strike wipes our your score completely. The waste bomb key enables players to detonate depth charges in order to launch another if he has missed. User defined graphics are extensively used. When typing in the program ensure that no unnecessary spaces are left, because the whole memory is needed.

```
POKE 16552,0(N/L)
POKE 16553,0(N/L)
```

or else the computer will clear display and draw again. The program returns to Basic when is pressed any key. The Subroutine at address 4082h — 16514 — gives — in C register — the frequency of the tone on the ear socket.

```
4082: 01 00 ff          4099: 4a
      db fe                    cb 38
```

3c	cb 38
20 01	d5
0c	cd b2 0b
10 f8	d1
c9	14
408e: 16 00	d5
cd 82 40	cb 72
2a 25 40	c4 2a 0a
2c	d1
c0	cb b2
41	18 e1

```
  1 POKE36878,15
  6 POKE55,18:POKE56,28
 10 PRINT"{CLR}WAIT":LL=0
 30 FORA=7168TO7679:POKEA,PEEK(A+25600):NEXT
 50 FORA=7432TO7503:READB
 70 POKEA,B:NEXTA
 81 GOSUB800:POKE0,197:POKE1,0:POKE673,0:POKE674,11:POKE675,33:POKE676,34:GOTO11
00
 90 DATA3,1,7,23,255,127,63,31,192,128,224,232,255,254,252,248
110 DATA1,1,5,31,63,63,63,31,224,224,224,254,255,255,255,254
130 DATA255,0,0,0,0,0,0,0,42,28,42,0,0,8,28,28,28,60,254,254,254,124
137 DATA0,1,30,32,67,68,137,137,240,12,228,18,17,145,81,81
138 DATAAC,A2,02,A9,20,99,2B,1E,A9,20,99,2C,1E,A2,00,A1,00,CD,A3,02,D0,07,C0,02
,F0,03
139 DATA88,A1,00,CD,A4,02,D0,05,C0,14,F0,01,C8,A9,21,99,2B,1E,A9,22,99,2C,1E,8C
,A2,02,60
140 CLR:POKE36879,10:POKE36869,255:PRINT"{CLR}{WHT}"
141 PRINT"{BLU}                "  :SYS680
143 A$(0)="228122312152221522151219122312251228222822283π"
145 FORA=1TO50STEP4:B=VAL(MID$(A$(0),A,3)):C=VAL(MID$(A$(0),A+3,1)):E=50:IFC=2T
HENE=100
146 IFC=3THENE=200
147 POKE36875,B:FORD=1TOE:NEXTD:POKE36875,0:NEXT
150 CLR:V=36875:SS=8185
160 X=7834:L=X
170 GOSUB500:TI$="000000"
180 GET A$:SYS680
181 PRINT"{HOME}{13 DOWN}TIME "MID$(TI$,4,1)","RIGHT$(TI$,2)
182 Z0=INT(RND(8)*21):IFZ0=3THENGOTO930
183 IFTI$>"000200"THENFORA=255TO128STEP-.1:POKEV+1,A:NEXT:POKEV+1,0:GOTO1000
184 IFA$=""THENIFX=L+(F*22)-23THENPOKEX+2,32:POKEX+1,32:GOSUB500
185 IFLL=0THENIFA$=".":THENLL=5:KK=0:GOSUB900:GOSUB700:POKEH,32:H=H+44
186 IFLL=0THENIFA$="."THENLL=5:GOSUB700:KK=3:GOSUB900:GOSUB700:POKEH,32:H=H+44
195 SYS680:IFLL<>5THEN230
200 IFA$="/"THENLL=5:SC=SC-2:POKEH,32:FORI9=1TO30:POKEV+2,210:NEXT:POKEV+2,0
230 POKEX,35:POKEX+1,36:POKEX-1,32:POKEX+2,32
240 SYS680:X=X+AZ:IFX=L+(F*22)+21THEN POKEX,32:POKEX-1,32:GOSUB500 -
245 SYS680:POKEV+1,240:POKEV+1,0
246 IFLL=5THENH=H+22:POKEH,38:IFPEEK(H-1)<>32ORPEEK(H+1)<>32THENPOKEH-22,32:GOS
UB600
247 IFLL=5THENIFH=<8120THENPOKEH-22,32
```

SOFTWARE FILE

Organisation

Andrew Dixon-Symes
West Croydon,
Surrey.

ORIC

THIS PROGRAM, for any size of Oric, turns the machine into a three-voice organ. It also shows how the play commands can be used to turn the certain voices on and off. The instructions are included in the program, and stay on screen while the program is running.

The second short program for Oric, demonstrates the high-resolution graphics of the machine whilst using the pattern command to give the lines being drawn a lace effect.

```
10 REM **** Three Voice Synthesizer ****
20 REM **** Copyright 2/3/1983     ****
30 REM ****    Andrew Dixon-Symes   ****
40 GOSUB 520
50 :
60 PLAY7,0,0,0
70 K$=KEY$
80 :
90 REM ****  VOICE ONE  ****
100 :
110 IFK$="1"THENMUSIC1,2,1,4
120 IFK$="2"THENMUSIC1,2,3,4
130 IFK$="3"THENMUSIC1,2,5,4
140 IFK$="4"THENMUSIC1,2,6,4
152 IFK$="5"THENMUSIC1,2,8,4
160 IFK$="6"THENMUSIC1,2,10,4
170 IFK$="7"THENMUSIC1,2,12,4
180 :
190 REM  ****  VOICE TWO  ****
200 :
210 IFK$="Q"THENMUSIC2,3,1,4
220 IFK$="W"THENMUSIC2,3,3,4
230 IFK$="E"THENMUSIC2,3,5,4
240 IFK$="R"THENMUSIC2,3,6,4
250 IFK$="T"THENMUSIC2,3,8,4
260 IFK$="Y"THENMUSIC2,3,10,4
270 IFK$="U"THENMUSIC2,3,12,4
280 :
290 REM  ****  VOICE THREE  ****
300 :
310 IFK$="A"THENMUSIC3,4,1,4
320 IFK$="S"THENMUSIC3,4,3,4
330 IFK$="D"THENMUSIC3,4,5,4
340 IFK$="F"THENMUSIC3,4,6,4
350 IFK$="G"THENMUSIC3,4,8,4
360 IFK$="H"THENMUSIC3,4,10,4
370 IFK$="J"THENMUSIC3,4,12,4
380 :
390 REM **** CONTROL PANEL ****
400 :
410 IFK$="Z"THENPLAY0,0,0,0
420 IFK$="X"THENPLAY1,0,0,0
430 IFK$="C"THENPLAY2,0,0,0
440 IFK$="V"THENPLAY4,0,0,0
450 IFK$="B"THENPLAY3,0,0,0
460 IFK$="N"THENPLAY5,0,0,0
470 IFK$="M"THENPLAY6,0,0,0
480 IFK$="L"THENPLAY7,0,0,0
490 IFK$=" "THENPLAY0,0,0,0:STOP
500 GOTO70
510 :
520 REM   **** INSTRUCTIONS ****
530 :
540 CLS
```

```
248 IFLL=5THENIFH=>8120THENLL=0:POKEH,32:POKEH-22,32
250 GOTO180
500 X=L:F=INT(RND(2)*12):AZ=INT(RND(3)*2):IFAZ=1THENAZ=-1
505 IFAZ=0THENAZ=1
510 X=X+(F*22):IFAZ=-1THENX=X-3
520 RETURN
600 POKEV+2,200:O=32
610 FORJJ=1TO10:POKEV-10,36:FORI9=1TO60:NEXT:POKEV-10,38:SYS680
611 FORI9=1TO60:NEXT:POKEV+2,0:POKEX+2,0:POKEX-1,0:POKEX+1,0:POKEX,0
620 IFAZ=1THENLL=0:POKEH,0:POKEH-22,0:SC=SC+10:GOSUB500:GOSUB700:GOTO630
621 IFAZ=-1THENLL=0:POKEH,0:POKEH-22,0:SC=SC+10:GOSUB500:GOSUB700
630 POKESS,39:SS=SS-1:GOTO180
700 PRINT"    SCORE    ▮▮▮▮"SC:RETURN
800 FORHH=680TO680+52:READ ZZ$
810 GG=ASC(ZZ$)-48:II=ASC(RIGHT$(ZZ$,1))-48
830 IFGG>16THENGG=GG-7
840 IFII>16THENII=II-7
850 POKEHH,GG*16+II
860 NEXT HH:RETURN
900 FORI9=7724TO7724+21
910 IFPEEK(I9+1)<>32THENH=I9:H=H+KK:POKEH,38:RETURN
920 NEXT:RETURN
930 I9=0:OO=250:FORXX=X-44TO7746STEP-22
935 SYS680
940 POKEXX,42:POKEV,170:POKEV+2,00:OO=OO-8:POKEXX+22,32
950 IFPEEK(XX-22)=33ORPEEK(XX-22)=34THENI9=5
960 IFI9=5THENSC=0:POKEXX-22,40:POKEXX-21,41:FORJJ=255TO128STEP-.5:POKEV,JJ:NEXT
961 IFI9=5THENPOKEXX-22,32:POKEXX-21,32:POKEV+2,0:POKEV,0:I9=0
970 NEXT:POKEV+2,0:POKEV,0
980 POKEXX,32:PRINT"                                  ":GOSUB700:GOTO183
1000 POKEV+4,27:POKEV-6,240
1010 PRINT"  YOU SCORED"SC" POINTS":PRINT" ANOTHER GO"
1030 PRINT"  (Y/N)▮"
1040 GETA$:IFA$=""THEN1040
1050 IFA$="Y"THEN140
1060 IFA$="N"THENEND
1070 GOTO1040
1100 PRINT"        THIS IS SUB HUNT ▮"
1110 PRINT"  YOU ARE A SHIP TRYINGTO STOP ENEMY SUBS    REACHING THEIR TARGET"
1115 PRINT"  BUT THEY FIRE BACK !"
1116 PRINT"  YOU HAVE 2 MINUTES"
1120 PRINT"  YOUR CONTROLS ARE:▮"
1130 PRINT"  Z FOR LEFT":PRINT"  C FOR RIGHT"
1140 PRINT"  ▮,▮ AND ▮ FOR FIRE":PRINT"  ▮ TO WASTE BOMB"
1150 PRINT"  HIT ANY KEY TO PLAY "
1160 GETA$:IFA$=""THEN1160
1170 GOTO140
```

```
550 PLOT10,2,"THREE VOICE SYNTHESIZER"
560 PLOT10,3,"----------------------"
570 A$=CHR$(96)+"Andrew Dixon-Symes:1983"
580 PLOT7,4,A$
590 PLOT3,7,"*** INSTRUCTIONS ***"
600 PLOT1,9,"VOICE-1:Numbers 1-7 In Octave 2"
610 PLOT1,10,"VOICE-2:Letters Q-U In Octave 3"
620 PLOT1,11,"VOICE-3:Letters A-J In Octave 4"
630 PLOT4,13,"The letters on the bottom row"
640 PLOT4,14,"act as on-off switches for"
650 PLOT4,15,"the different combinations of"
660 PLOT4,16,"the three sound channels."
670 PLOT5,17,"Z=ALL OFF,X=1 ON,C=2 ON,"
680 PLOT5,19,"V=3 ON,B=1&2 ON,N=1&3 ON,"
690 PLOT5,21,"M=2&3 ON,L=1,2&3 ON."
700 PLOT1,23,"TO STOP THE PROGRAM PRESS THE SPACE BAR"
710 RETURN
```

```
10 REM ****  ORIC PATTERN  ****
20 REM **** Copyright 1983 ****
30 REM ****    Andrew D-S   ****
40 :
50 HIRES
60 :
70 A=INT(RND(1)*90)+100
80 B=INT(RND(1)*60)
90 C=INT(RND(1)*3)-4
100 :
110 REPEAT:Z=Z+1
120 :
130 FOR R=A TO B STEP C
140 :  CURSET R,R,3
150 :  DRAW 0,199-2*R,1
160 :  DRAW 239-2*R,0,1
170 :  DRAW 0,-199+2*R,1
180 :  DRAW -239+2*R,0,1
190 :  PATTERN INT(RND(1)+127)+127
200 NEXT R
210 :
220 UNTIL Z=3
230 WAIT 200
240 Z=0:GOTO 50
```

Bubbles

D J Berry,
Rogerstone,
Gwent.

DRAGON

BUBBLES IS A game written entirely in Dragon Colour Basic. Listing 1 is the game itself, listing 2 is a separate instructions program. You can read the instructions now to get an idea of how to play the game.

The program relies heavily on the speed of Dragon Basic graphics commands for its fluidity. Circles are drawn, un-drawn and moved to provide targets for a laser gun which is initially drawn with the Draw command and subsequently moved with Put and Get. Because I was limited to Inkey$ for movement control the gun moves continuously — key presses modify rather than initiate movement. The only exception to this is in line 60 where Peeking location 65288 reads keyboard-matrix columns to detect when the gun is to be fired.

The graphics commands most often used in the program are: Circle, Draw, Put and Get. Circle, as the command name implies, draws a circular form on the screen centred at X,Y and with defined radius, colour and height to width ratio — unity for a circle, other values for ellipses. The resolution of the circle is far better in the highest graphics mode

```
10 'LISTING 2 -- INSTRUCTIONS
20 '************************
30 CLS
40 PRINT STRING$(32,"=");
50 PRINT"** WELCOME TO DRAGON BUBBLES **"
60 PRINT STRING$(32,"=");
70 PRINT"A DEADLY VIRUS IS FLOATING DOWN"
80 PRINT"FROM THE SKIES INSIDE GOSSAMER"
90 PRINT"BUBBLES. YOUR TASK AS LASER"
100 PRINT"COMMANDER IS TO SHOOT  ALL THE"
110 PRINT"BUBBLES BEFORE THEY REACH THE"
120 PRINT"GROUND. ONLY ONE HAS TO TOUCH"
130 PRINT"DOWN FOR THE WHOLE WORLD TO BE"
140 PRINT"WIPED OUT."
150 IF INKEY$="" GOTO 150
160 CLS
170 PRINT STRING$(32,"=");
180 PRINT"** WELCOME TO DRAGON BUBBLES **"
190 PRINT STRING$(32,"=");
200 PRINT"UNFORTUNATELY EACH TIME ONE IS"
210 PRINT"HIT TWO THINGS HAPPEN.  FIRST"
220 PRINT"THAT BUBBLE GETS SMALLER, THEN"
230 PRINT"IT BEGINS TO FALL FASTER."
240 PRINT"ONE OTHER PROBLEM IS THAT LASER"
250 PRINT"CANNON CAN'T FIRE THROUGH CLOUD":PRINT
260 PRINT"ALSO, AS TIME GOES ON, THE"
270 PRINT"SKIN OF THE BUBBLES BECOMES"
280 PRINT"MUCH TOUGHER, REQUIRING MUCH"
290 PRINT"MORE ACCURATE SHOOTING."
300 IF INKEY$="" GOTO 300
```

```
310 CLS
320 PRINT STRING$(32,"=");
330 PRINT"** WELCOME TO DRAGON BUBBLES **"
340 PRINT STRING$(32,"=");
350 PRINT"TO CONTROL THE CANNON USE THE"
360 PRINT"FOLLOWING KEYS:"
370 PRINT STRING$(03," ");"< -- TRAVERSE LEFT FAST"
380 PRINT STRING$(03," ");"> -- TRAVERSE RIGHT FAST"
390 PRINT STRING$(03," ");"M -- TRAVERSE LEFT SLOW"
400 PRINT STRING$(03," ");"/ -- TRAVERSE RIGHT SLOW"
410 PRINT STRING$(03," ");"(SPACE BAR) -- FIRE CANNON"
420 IF INKEY$="" GOTO 420
430 CLS
440 PRINT STRING$(32,"=");
450 PRINT"** WELCOME TO DRAGON BUBBLES **"
460 PRINT STRING$(32,"=");
470 PRINT"THE GAME SCORES ARE DISPLAYED"
480 PRINT"AT THE TOP RIGHT OF THE  SCREEN"
490 PRINT"IN THE  SEQUENCE: GAME  NUMBER"
500 PRINT"NUMBER OF BUBBLES LEFT IN THIS GAME),"
510 PRINT"SCORE (DEPENDS ON RADIUS  AND"
520 PRINT"ACCURACY), AND TOTAL  NUMBER OF"
530 PRINT"BUBBLES DESTROYED."
540 IF INKEY$="" GOTO 540
550 FOR A=8 TO 1 STEP -1
560 CLS(A)
570 TIMER=0
580 PRINT @160," "
590 PRINT @170,"GOOD LUCK"
600 IF TIMER<100 GOTO 600
610 NEXT
```

SOFTWARE FILE

```
10 'LISTING 1 -- BUBBLES -- D.BERRY
20 '******************************
30 CLS8
40 XGUN=6:MOVE=-5:HIT=0:DAMAGE=0
50 QUANTITY=5:RAD=15:ACC=1:GAME=1
60 DIM MB(8,8),X(QU),Y(QU),VEL(QU),RAD(QU),GUN(10,12),BLANK(10,12),Q$(4),CLOUD(26,8),BCLOUD(26,8)
70 REM PARAMETERS:
80 C0=65288:C1=223:C2=-5:C3=+5:C4=-2:C5=+2:C6=178:C7=190:C8=240:C9=157:K0=158:K1=150:K2=26
90 Q$(1)="BM206,2:"
100 Q$(2)="BM211,2:"
110 Q$(3)="BM216,2:"
120 Q$(4)="BM221,2:"
130 PMODE4,1:SCREEN1,0:COLOR 0,1:PCLS
140 GET(0,0)-(26,8),BCLOUD,G
150 DRAW"BM5,5:F2R16E2U1H3L6G1L4H4L5G4D2R6"
160 PAINT(10,6)
170 GET(0,0)-(26,8),CLOUD,G
180 PUT(0,0)-(26,8),BC,PSET
190 GET(20,179)-(28,187),MB,G
200 LINE(0,0)-(255,191),PSET,B
210 PLAY"T10:ABCDEDCBA:T255"
220 TTL=QU+1
230 GOSUB 310
240 GOSUB 420
250 GOSUB 1170
260 GOSUB 1040
270 GOSUB 650
280 GOSUB 470
290 IF TTL>1 GOTO 270
300 IF TTL=1 THEN GA=GA+1:RAD=RAD*0.75:MOVE=-5:XGUN=3:ACC=ACC-0.25:IF ACC<0.5 THEN ACC=0.5:PCLS:GOTO 200 ELSE PCLS:GOTO 200 ELSE PLAY"T5:03DDD02B":PAINT(2,190):CLS(GA):PRINT"YOU LOST":PRINT"YOU SCORED  ":PRINT FIX(DA)" POINTS":PRINT HIT;"DESTROYED":END
310 REM DCIRC:DRAW CIRCLES
320 XCLOUD=0
330 FOR Q=1 TO QU
340 X(Q)=RND(255-2*RAD)+RAD
350 Y(Q)=RND(48-RAD)+RAD
360 RAD(Q)=RAD
370 VEL(Q)=15/RAD
380 IF X(Q)>50 AND X(Q)<150 AND XCLOUD=0 THEN XCLOUD=X(Q)-12
390 NEXT Q
400 IF XCLOUD=0 THEN XCLOUD=175
410 RETURN
420 REM DGUN:DRAW GUN
430 DRAW"BM3,190:R09U1L09U1R09U1L09:BM7,186:U8R1D8"
440 GET(3,178)-(13,190),GUN,G
450 GET(20,174)-(30,186),BLANK,G
460 RETURN
470 REM MCIRC:MOVE CIRCLES
480 TIMER=0
490 PUT(XC,K1)-(XC+K2,K0),BC,PSET
500 XC=XC-(RND(3)-2)
510 PUT(XC,K1)-(XC+K2,K0),CL,AND
520 RN=RND(QU)
530 IF RAD(RN)<1 GOTO 630
540 CIRCLE(X(RN),Y(RN)),RAD(RN),1
550 PSET(X(RN),Y(RN))
560 Y(RN)=Y(RN)+VEL(RN)
570 IF Y(RN)>180-RAD(RN) GOTO 300
580 X(RN)=X(RN)+(RND(7)-4)
590 IF X(RN)<10 THEN X(RN)=10
600 IF X(RN)>243 THEN X(RN)=243
610 CIRCLE(X(RN),Y(RN)),RAD(RN),0
620 PSET(X(RN),Y(RN))
630 IF TIMER<16 THEN GOTO 630
640 RETURN 'TIMER CONTROLS SMOOTH GUN MOVEMENT
650 REM MGUN:MOVE GUN
660 IF PEEK(C0)=C1 THEN PLAY"C":GOSUB 850
670 MG$=INKEY$
680 IF MG$="" GOTO 730
690 IF MG$="," THEN MOVE=C2
700 IF MG$="." THEN MOVE=C3
710 IF MG$="M" THEN MOVE=C4
720 IF MG$="/" THEN MOVE=C5
730 PUT(XGUN,C6)-(XGUN+10,C7),BLANK,PSET
740 XGUN=XGUN+MOVE
750 IF XGUN>C8 OR XGUN<C3 THEN MOVE=-MOVE
760 IF XGUN<0 THEN XGUN=6
770 PUT(XGUN,C6)-(XGUN+10,C7),GUN,PSET
780 RETURN
790 REM FIRE:FIRE GUN
800 TIMER=0
810 LINE(0,C6)-(0F,YGUN),PSET
820 PLAY"D"
830 LINE(0F,C6)-(0F,YGUN),PRESET
840 RETURN
850 REM HIT? -- DRAW OUT
860 YMAX=0:HTEST=0:0F=XGUN+4
870 FOR Q=1 TO QU
880 IF RAD(Q)=0 GOTO 920
890 IF(0F+(RAD(Q)*ACC)>X(Q)) OR (0F-(RAD(Q)*ACC)>X(Q)) GOTO 920
900 HTEST=1
910 IF Y(Q)>YMAX THEN YMAX=Y(Q):K=Q
920 NEXT
930 IF HTEST=0 THEN YGUN=1 ELSE YGUN=Y(K)
940 IF 0F>XC THEN IF XG<XC+22 THEN IF YG<C9 THEN YG=K0:GOSUB 790:RETURN
950 GOSUB 790
960 IF YGUN=1 THEN RETURN
970 REM RUB OUT & REDRAW
980 CIRCLE(X(K),Y(K)),RAD(K),1
990 RAD(K)=FIX(RAD(K)/2)
1000 VEL(K)=VEL(K)*2:X(K)=X(K)+RND(7)-4
1010 IF RAD(K)>0 THEN CIRCLE(X(K),Y(K)),RAD(K),0 ELSE HIT=HIT+1:PRESET(X(K),Y(K)):X(K)=0:TTL=TTL-1:GOSUB 1170
1020 DA=DA+(VEL(K)/ACC):GOSUB 1040
1030 RETURN
1040 REM DCOUNT:PRINT DAMAGE TALLY
1050 PUT(206,2)-(214,10),MB,PSET
1060 PUT(216,2)-(224,10),MB,PSET
1070 DX=DA:SCALE=1000
1080 FOR LOOP=1 TO 4
1090 AA$=Q$(LO)
1100 BB=FIX(DX/SCALE)
1110 GOSUB 1350
1120 DX=DX-BB*SCALE
1130 SCALE=SCALE/10
1140 NEXT
1150 PLAY"C"
1160 RETURN
1170 REM HCOUNT:HIT TALLY
1180 AA$="BM193,2:"
1190 PUT(193,2)-(201,10),MB,PSET
1200 BB=TTL-1
1210 GOSUB 1350
1220 AA$="BM181,2:"
1230 PUT(181,2)-(189,10),MB,PSET
1240 BB=GAME
1250 GOSUB 1350
1260 AA$="BM236,2:"
1270 PUT(236,2)-(244,10),MB,PSET
1280 BB=FIX(HIT/10)
1290 GOSUB 1350
1300 AA$="BM241,2:"
1310 BB=HIT-(FIX(HI/10))*10
1320 GOSUB 1350
1330 PLAY"ADA"
1340 RETURN
1350 REM PRINT:PRINT ON SCREEN
1360 ON BB GOTO 1380,1390,1400,1410,1420,1430,1440,1450,1460
1370 AB$="R3D7L3U7":GOTO 1470
1380 AB$="BM+3,0:D7":GOTO 1470
1390 AB$="R3D3L3D4R3":GOTO 1470
1400 AB$="R3D7L3R3U4L3":GOTO 1470
1410 AB$="D3R3U3D7":GOTO 1470
1420 AB$="R3L3D3R3D4L3":GOTO 1470
1430 AB$="R3L3D7R3U4L3":GOTO 1470
1440 AB$="R3D7":GOTO 1470
1450 AB$="R3D3L3U3":GOTO 1470
1460 AB$="R3D3L3U3R3D7":GOTO 1470
1470 DRAW AA$+AB$
1480 RETURN
```

(continued from previous page)

PMode4. In the lower modes the circles are approximate to such an extent that with radii below about 15 their finned square shapes make useful gunsights. The Draw command is always followed by a string literal or variable which controls the movement of an imaginary cursor — up, down, left, right; plus diagonals, rotation, scaling and blank moves any number of screen points.

The gun and cloud are formed initially with the Draw command then moved around with Put and Get. The score digits are produced by concatenating strings containing draw instructions with ones containing blank move instructions then using the result as the argument for a Draw command.

Get copies graphical data from the screen and stores it in a predefined array. Put reverses the process. Put and Get are used in Bubbles to move the gun and cloud, the sequence being simply: Put blank sky where the gun is now; change the X co-ordinate of the gun; Put the gun in its new place. the cloud is moved in the same way.

The program is broken down into a series of subroutines starting with DCirc in line 310.

Lines 10 to 220 perform the initial setting up, draw the screen, and cloud, set tally and score counters and set up a parameter list.

The arrays used by the program are as follows: MB stores a bit of blank sky used here and there to rub out characters and so on, X and Y are circle centres, Vel is circle 'velocity'', Rad is radius, Gun has the drawn "gun" Put and Get from it, Blank is again blank sky — this time big enough to rub out the gun, and Cloud and BCloud are similar to Gun and Blank. It is important to note that Dragon Basic only recognises the first two letters of a variable name so that, for example, Bcloud and BC are the same variable. Lines 150 and 160 use Draw and Paint to fashion the nasty black cloud.

DCirc sets up each circles' variables: X,Y,Rad and Vel. It also ensures the cloud is directly in front of one of the bubbles. DGun draws the gun and stores it in array Gun.

MCIRC first moves the cloud then draws out, moves and redraws a randomly chosen circle. Line 570 asks "has a bubble reached the ground?" then jumps to a line which terminates the program if true.

MGun scans the keyboard and jumps to Hit if the fire key is being pressed. Notice the extensive use of parameters in this often used routine. Fire simply draws the line of "laser light" from "gun" to "target". This is called from Hit which calculates whether a bubble was in the way of the shot. It uses the present Accuracy and Radius values to perform the calculation. This routine produces a parameter YGun to pass into Fire which then draws its line to the point XG,YG. Lines 970 onwards change the parameters of a hit bubble. Radius is halved velocity doubles and its centre jumps sideways.

DCount and HCount are respectively damage and hit tallies. They break the number passed to them into individual digits then use routine Print to display the values on the screen.

Worm squirm

J S Henry,
Cranleigh,
Surrey.

ATOM

THIS PROGRAM is called Worm and it runs on a fully expanded Acorn Atom in 4K. The

A two-minute operation turns your BBC Micro into the heart of a word processor.

VIEW is a software program from Acornsoft (the software division of Acorn Computers Ltd., who designed and built the BBC Micro) that enables you to use your BBC Micro, together with a printer, as a fully operational word processor.

View is supplied as a Rom chip that can easily be fitted to your BBC Micro by your local dealer, in a painless two-minute operation.

Then, once installed, you only have to switch on and View is operating immediately. (You can easily switch back to normal computing with a single command.)

Also included in the View package are two special books: 'Into View,' that takes you by easy stages through all the word processing commands and explains the many ways in which View can help you, and the 'View Guide,' which provides a quick reference to all View facilities.

You'll find that View is, by any standards, a thoroughly professional system, yet still surprisingly simple for the beginner to master.

The 'Spark-Jet Printer' shown in the photograph is the ideal choice of printer for your word processing application. Extremely quiet, it offers high resolution graphics from monitor or T.V. screen and is available now from dealers.

If you'd like more information, write to Acornsoft, 4a Market Hill, Cambridge CB2 3NJ.

Or, for details of your local Acornsoft dealer, phone 01-200 0200.

ACORNSOFT

TERROR-DAKTIL 4D

In a place forgotten by time, the beating of wings heralds an incarnation too horrifying to believe – Terror Daktil 4D. A game so terrifyingly real that you'll need all your courage just to remain at your 48K Spectrum screen. As the dreaded prehistoric monsters swoop down, intent on your destruction, a battered pistol and marksmanship are your only defence.

You must fight for your life and as the sun sets, your chances of survival begin to slip away. For at night, you'll only see the gleam of their eyes!

Terror Daktil 4D – a game beyond your wildest dreams; not just 3-dimensional graphics reaching out of your Spectrum to engulf you, but also the 4th dimension of time that turns day into menacing night at an alarming rate.

Unbelievable thrills await those brave enough to enter the lost world of Terror Daktil 4D.

Two more super 48K Spectrum programs;

PENETRATOR "...immensely playable, yet very difficult and horribly addictive." – **Computer & Video Games.**

THE HOBBIT "The Hobbit is far superior to any other adventure game available for the Spectrum" – **Your Computer** "A game by which future games will be judged." – **P.C.T.**

MELBOURNE HOUSE PUBLISHERS

object of the game is to eat the zeros and miss the dots. There are always two zeros on the screen and when you eat one, another one appears.

If you hit one of the dots a section of your tail is cut off and another dot appears. Dots also appear after you have eaten 3 zeros. If you run into your tail or the sides you will lose a point. The game ends when you either run out of tail or trap yourself in your tail. The dots move randomly one at a time and there are machine-code sound effects.

Use j to turn you left and k to turn you right. To master the turning always imagine yourself at the head of the worm looking the direction

you are always travelling. Make sure also that you do not keep the button down too long or you will turn too far. Here is a breakdown of the program:

```
    1-95    : set up screen and machine code
  100-160   : move worm
```

```
1000-1030 : set up zero
1100-1120 : set up dots
1200-1250 : check if dead end or trapped
1290-1370 : game over routine
1500-1530 : move a dot at random
1999-end  : instructions.
```

```
160S.r
21#0;E=0
57#81=3;I=0;R=#3BF0;$R="A";2
10?#E1=0;CLEAR0;S=1;IFK;IORH>9999;H=0
20MOVE0,0;DRAW63,0;DRAW63,47;DRAW0,47;DRAW0,0
30DIMAA21,EE20,VV5;P=#3B00;P,#21
40L=#B002;C,:VV0JSR#FE71;STY#80;RTS
50:VV1LDY#81;:VV2LDX#80
60:VV3DEX;BNEVV3
70STAL;EOR#4;DEY;BNEVV2;RTS;J
80P.16;H=15
90FORG=1TOM;AAG=#3020+G;?AAG=-1;N.
91FORG=255TO1S.-4;?#80=G;LINKVV1;WAIT;N.
9560S.a;?#81=20
100?AAM=162
110LINKVV0;Z=?#80+32;?#B002=0
1150S.x
```

```
1201I+(Z=CH"K")-(Z=CH"J");I=(I+4)%4
1300=AAM+R?I-64;IF?Q<>64;G0S.c
1351IFT=1;T=0;G.100
1401AAC(M+1)=0;?AA(M+1)=162;?AAM=-1
145?AA1=64
150FORG=1TOM;AAG=AA(G+1);N.
160?#80=Z*3;LINKVV1;G.110
1000AG=#3000+R.%480;IF?G<>64;G.a
1010?G=15;FORG=1TO10;?#B002=?#B002%7;N.
1020S=S+1;IFS%3=0;G0S.b
1030R.
1100bG=#3000+R.%480;IF?G<>64;G.b
1105E=E+1;EEE=G
1110?G=46;?#80=40;FORG=1TO10;LINKVV1;WAIT;N.
1120R.
1200cIF?Q=15;G0S.a;R.
```

```
1210IFAAM?1>64;IFAAM?-1>64;IFAAM?32>64;IFAAM?-32>64;G.d
1215T=0
1220IF?Q=46;G0S.e;IFH=0;G.d
1230FORG=0TO255S.36;?#80=G;LINKVV1;N.;T=1
1240IFRND%4=0;F.G=1TO10;?#80=RND;LINKVV1;N.;S=S-1;R.
1250R.
1290d?#81=2
1300CLEAR4;FORG=1TO200;?#B000=#F0;?#80=G;LINKVV1;?#B000=#D0;N.
1305P.#12"game over"'''"YOUR SCORE WAS"S'':IFH<S;H=S
1310P."HISCORE"'       "H"'
1320FORG=255TO0S.-8;?#80=G;LINKVV1;N.
1330?#81=20;FORG=1TO255S.3
1340?#80=G;LINKVV1;?#80=G-10;LINKVV1;?#80=G+10;LINKVV1
1350N.;P."PRESS ANY KEY TO RESTART"
1360?#80=RND;?#81=3;LINKVV1;LINKVV0;IF?#80<255;G.2
```

```
1370G.1360
1400c?AAM=64;FORG=1TO10
1410?#80=G;LINKVV1;?#80=255-G;LINKVV1;N.;M=M-1;R.
1500x0=R.R.%E+1
1510G=EE0+R.%2+R.%2*32
1520IF?G<>64;R.
1530?EE0=64;?G=46;EE0=G;R.
1999r
2000P.#12"************************NO7************"
2010P.'''"YOU ARE AT THE HEAD OF THE WORM."''"YOU MUST EAT THE "
2020P."'0'S.IF YOU HIT THE DOTS YOU WILL LOSE PART OF YOUR TAIL"
2030P.'.MAKE SURE YOU DON'T RUN INTO YOUR TAIL OR THE WALLS OR"
2040P.'" YOU MIGHT TRAP YOURSELF "'
2050P."USE 'J' TO MOVE LEFT AND 'K' TO MOVE RIGHT"'
2060P.'"GOOD LUCK"'''"PRESS ANY KEY TO CONTINUE";L1.#FFE3
2070R.
```

Flight of Fancy

M J Parrot,
Stockport,
Cheshire.

BBC

THE GAME is for the Model B and is the traditional 501 down, finishing on a double or the bull. Lines 10,220 give the instructions for the game, ask the number of players and take in the names of the players. If there is only one, then player 2 is assigned the name Beeb (line 210) and the logical variable Beeb is set True.

After pressing the space bar (lines 230,240) the game can begin but we first have to draw the board. This is the most complicated part of the program because some trigonometry is used. Line 1170 first sets minimum and maxi-

mum values for X and Y so that the cross-hair cannot wander off the edge of the screen. The two variables Dif and Mote are used to draw the cross-hair and move it around the board. Line 1180 Reads the scores for each sector of the board into an array S in a counter-clockwise fashion beginning and ending with the 6 (at three o'clock). The origin of the graphics page is set to the centre of the screen and we are set for drawing sectors and circles around this central point. The radii which delimit the bull, the outer bull, the trebles and the doubles are set in line 1200 and two variables C and S are set to the cosine and sine respectively of 0.02 of Pi because the sectors are drawn recursively to save time. I used the triangle fill mode to draw five pairs of triangles in each segment since this nicely fills the screen leaving a clear image without unlit pixels.

The main part of the board is drawn by a pair of nested For-Next loops. The outer loop — lines 1210-1280 — steps around the board in steps of 18 degrees.

The inner loop — lines 1230-1260 — draws four segments within each sector by calling Procedure Sector. Within this loop the colour is changed alternatively between the two logical, graphical foreground colours 0 and 1 by the line 1240 which uses the variable K which was initially set in line 1200. On leaving the inner loop the logical colour is once more changed to allow for the start of the next sector.

Procedure Sector Moves the graphics cursor to the start X,Y of the segment and Draws the leading edge up to the point X1, Y1. The third point of the triangle 1 is calculated recursively and the triangles are filled in. The next

```
10  MODE1:COLOUR2:COLOUR128:BEEB=FALSE
20  DIM S(5),S(20),SCORE(1),DARTX(3),DARTY(3),NAMES(1)
30  PRINTTAB(15,2)"BEEB-DARTS"
40  COLOUR3
50  PRINT:PRINT"The game is 501 down."
60  PRINT"Finish on a double or the bull."
70  COLOUR1
80  PRINT:PRINT"AIMING"
90  COLOUR3
100 PRINT:PRINT"Up.....A       Down.....?"
110 PRINT:PRINT"Left....K      Right....L"
120 PRINT:PRINT:PRINT"Throw with the space bar"
130 PRINTTAB(6,20)"Will there be 1 or 2 players"
140 REPEAT
150     A$=GET$
160     UNTIL A$="1"OR A$="2"
170 IFA$="1"THENBEEB=TRUE
180 PRINTTAB(6)"What's the name of player 1"
190 INPUTNAME$(0)
200 IF NOT BEEB THENPRINTTAB(6)"What's the name of player 2":INPUTNAME$(1)
210 IF BEEB THENPRINT"The BEEB will be player 2":NAME$(1)="BEEB"
220 PRINT:PRINT"PRESS THE SPACE BAR TO PROCEED"
230 REPEAT
240     A$=GET$:UNTILA$=" "
250 MODE1:GCOL0,130:CLG:COLOUR130:COLOUR3
260 PROCBOARD
270 FINISH=FALSE
280 REPEAT
290     FORPLAYER=0TO1:SCORE(PLAYER)=501:PROCWRITESCORE
300     IFLEN(NAME$(PLAYER))>6THENNAME$(PLAYER)=LEFT$(NAME$(PLAYER),6)
310     NEXT
320     PLAYER=1
330     GAMEOVER=FALSE
340     REPEAT
350         PLAYER=ABS(PLAYER-1):DART=0:TSCORE=0
360         PROCWRITENAME
370         REPEAT
380             DART=DART+1
390             DOUBLE=FALSE:BUST=FALSE:WIREFLAG=FALSE
400             X=RND(470)-235:Y=RND(470)-235
410             PROCMOVECROSS
420             IFPOINT(X,Y)=2AND RND(7)=2THEN WIREFLAG=TRUE
430             PROCDART(X,Y)
440             *FX15,1
450             PROCTHUD
460             PROCSCORE
470             IFR<R(1) OR R>R(5) THENWIREFLAG=FALSE
480             PROCWRITESUB
490             PROCTOTALISE
500             UNTIL DART=3 OR GAMEOVER OR BUST
510         IF NOT BUST THENSCORE(PLAYER)=SCORE(PLAYER)-TSCORE
520         PROCWAIT
530         *FX15,1
540         PROCCLEARBOARD
550         PROCWRITESCORE
560         UNTILGAMEOVER
570     PROCWRITEND
580     REPEATA$=GET$:UNTIL A$="N"OR A$="Y"
590     IFA$="N"THENFINISH=TRUE
600     PROCWRITEND
610     VDU4
620     UNTIL FINISH = TRUE
630 MODE1:PRINTTAB(18,10)"'BYE"
640 END
650 DEF PROCSCORE
660 
670 DARTX(DART)=X:DARTY(DART)=Y
680 R=SQR(X^2+Y^2):IFR<R(0)THENSCORE=50:DOUBLE=TRUE:ENDPROC
690 IFR<R(1)THENSCORE=25:ENDPROC
700 IFR>R(5)THENSCORE=0:ENDPROC
710 IFWIREFLAG THENSCORE=0:ENDPROC
720 THETA=ACS(X/R):IFY<0THENTHETA=2*PI-THETA
730 THETA=INT(0.7+THETA*10/PI-PI/20)
740 SCORE=S(THETA):IFR<R(4)THENSCORE=SCORE*2:DOUBLE=TRUE:ENDPROC
750 IFR<R(3)ANDR>=R(2)THENSCORE=SCORE*3:ENDPROC
760 ENDPROC
770 DEF PROCWRITESUB
780 
790 IFWIREFLAG THENPRINTTAB(1+PLAYER*34,25+DART)"wire"ELSEPRINTTAB(1+PLAYER*36,25+DART);SCORE
800 ENDPROC
810 
820 DEF PROCDART(X,Y)
830 D=DIF/2
840 MOVEX-D,Y-D
850 DRAWX+D,Y+D
860 MOVEX-D,Y+D
870 DRAWX+D,Y-D
880 ENDPROC
890 
900 DEFPROCMOVECROSS
910 GCOL3,2
920 REPEAT
930     PROCRANDOM
940     IFX<XMIN THENX=XMIN
950     IFX>XMAX THENX=XMAX
960     IFY<YMIN THENY=YMIN
970     IFY>YMAX THENY=YMAX
980     PROCCROSS
990     A$=INKEY$(1)
1000    PROCCROSS
1010    IFA$="A" Y=Y+MOTE
1020    IFA$="Z" Y=Y-MOTE
1030    IFA$="K" X=X-MOTE
1040    IFA$="L" X=X+MOTE
1050    IFBEEB AND PLAYER=1THENPROCMYGAME
1060    UNTILA$=" "
```

(listing continued on page 175)

ZX81 MACHINE CODE?

ZX.ASZMIC rom transforms ZX81 into an

Assembly Language programming unit

ø FULL-SCREEN EDITOR

Sixteen shift keys take you into a world with a word processor feel. A blink cursor moves at your command to control insertion, rubout, line or string deletion, atuoscroll & page flip up or down. Text block operations. Ulta-fast editing.

ø MULTI-FILE SYSTEM

Declare as many files as you like, with any names you like, & they are automatically handled by the Operating System. Merge them, delete them, print, save & load them, & edit them by name. Superb flexibility with a simple but powerful system.

ø TOTAL ASSEMBLER

Full Z80 mnemonics, unlimited length labels, ORG & EQU directives, proper assembly listings with errors flagged on screen or printer. Relocatable object code & options to faciliatate cross-assembly. Interpretive immediate execution available.

ø POWERFUL DEBUG
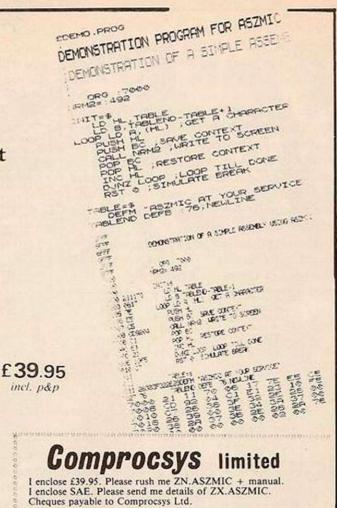
All the usual dump, modify, fill & copy commands; plus breakpoints, single stepping, context control, the convenience of interpretive execution mode, full use of the names in your program, Command Macros, autodump, and full operating system interface.

ø HI-RES GRAPHICS

255 × 144 resolution under program control to give you truly convincing graphics. With the power & flexibility of assembler you can really use this high definition.

ø MUCH, MUCH MORE

Repeat function on all keys. Double height & fine pitch on printer. Cassette catalogue. 120 page "ASZMIC GOLD" manual with training exercises, discussion chapters, sample programs, documented monitor routines & much reference information. Communicate with Basic via cassette (skeleton programs with machine code in REM) or via memory using the optional software switchable dual ROM board available from Capital Computers Ltd (£9 + vat). ASZMIC redefines the ZX81 hardware to give you an integrated development system with professional facilities which rival even very expensive systems. Why not get in touch for more details?

£39.95
incl. p&p

(listing continued from page 173)

```
1070 ENDPROC
1090 DEF PROCCROSS
1100 MOVEX-D1F,Y
1110 DRAWX+D1F,Y
1120 MOVEX,Y-D1F
1130 DRAWX,Y+D1F
1140 ENDPROC
1160 DEF PROCBOARD
1170 ZMIN=-630:XMAX=630:YMIN=-512:YMAX=512:D1F=50:HDIF=15
1180 FORI=1TODO:READS(I):NEXT
1190 VDU5:VDU29,640!512;
1200 R(0)=10:R(1)=30UL(2)=250:R(3)=280:R(4)=440:R(5)=470:C=COS(0!1/50):S=SIN(0!1/
50):I=1
1210 FORI=-PI/20TO2*PI-PI/20STEPPI/10
1220 C1=COS(I):S1=SIN(I)
1230 FORL=1TO4
1240    GCOL0,K:K=ABS(K-1)
1250    PROCSECTOR(R(L)*C1,R(L)*S1,R(L+1)*C1,R(L+1)*S1)
1260    NEXTL
1270 GCOL0,K:K=ABS(K-1)
1280 NEXTI
1290 FORI=0TO19:MOVE490*COS(I*PI/10)-300-25*((S(I)=18)*(S(I)=13)*(S(I)=10)+(S(I
)=15)),490*SIN(I*PI/10)*10
1300    PRINTS(I):NEXTI
1310 GCOL0,2
1320 FORI=-PI/20TOC2*PI-PI/20STEPPI/10
1330    X=R(5)*COS(I):Y=R(5)*SIN(I)
1340    MOVE0,0:DRAWX,Y
1350    NEXT
1360 FORI=2TO4:PROCCIRCLE(R(I)):NEXT
1370 GCOL0,0
1380 C=PI/10:MOVE0,0
1390 FORI=1TO21:X=R(0)*COS(C*(I-1)):Y=R(0)*SIN(C*(I-1))
1400    MOVE0,0:PLOT85,X,Y
1410    NEXTI
1420 VDU4
1430 ENDPROC
1450 DEF PROCSECTOR(X,Y,X1,Y1)
1460 MOVEX,Y:DRAWX1,Y1
1470 FORJ=1TO5
1480    T=X*C-Y*S:Y=Y*C+X*S:X=T
1490    PLOT85,X,Y
1500    T=X1*C-Y1*S:Y1=Y1*C+X1*S:X1=T
1510    PLOT85,X1,Y1
1520    NEXTJ
1530 ENDPROC
1550 DEF PROCTOTALISE
1560 IFSCORE=SCORE(PLAYER)-TSCORE ANDDOUBLE=TRUE THEN GAMEOVER=TRUE:ENDPROC
1570 IFSCORE>=SCORE(PLAYER)-TSCORE-1THENPROCBUST:ENDPROC
1580 TSCORE=TSCORE+SCORE
1590 ENDPROC
1610 DEF PROCBUST
1620 SOUND 1,-15,100,10
1630 BUST=TRUE
1640 ENDPROC
1660 DEF PROCWRITESCORE
```

```
1670 FLAG=FALSE:SCORE=SCORE(PLAYER)
1680 IFSCORE=41AND(SCORE/2-INT(SCORE/2))<>0,1THENSCORE=SCORE/2:FLAG=TRUE
1690 PRINTTAB(1+PLAYER*35,4)"    "
1700 PRINTTAB(1+PLAYER*35,4):SCORE;
1710 IFFLAG PRINT "2"
1720 ENDPROC
1740 DEF PROCCLEARBOARD
1750 FORI=1TODART
1760    PROCDART(DARTX(I),DARTY(I))
1770    PRINTTAB(1+PLAYER*34,25+I)"    "
1780 PROCWAIT
1790 NEXTI
1800 ENDPROC
1820 DEF PROCCIRCLE(R)
1830 C=COS(0.05):S=SIN(0.05):X=R:Y=0
1840 MOVEX,Y
1850 FORJ=1TO128:T=X*C-Y*S:Y=Y*C+X*S:X=T
1860    DRAWX,Y
1870    NEXTJ
1880 ENDPROC
1900 DEF PROCWAIT
1910 FORJ=1TO1000:NEXTJ
1920 ENDPROC
1940 DEF PROCMYGAME
1950 PROCWAIT
1960 A$=" "
1970 IF SCORE(PLAYER)>180THENX=0:Y=R(2)+10:PROCRANDOM:ENDPROC
1980 IF SCORE(PLAYER)>100THENX=0:Y=R(3)+25:PROCRANDOM:ENDPROC
1990 IF SCORE(PLAYER)-TSCORE>61THENX=30:Y=R(3)+30:PROCRANDOM:ENDPROC
2000 IFSCORE(PLAYER)-TSCORE>50THENX=0:Y=5:PROCRANDOM:ENDPROC
2010 IF SCORE(PLAYER)-TSCORE>40THENX=R(3)+30:Y=0:PROCRANDOM:ENDPROC
2020 SCORE=(SCORE(PLAYER)-TSCORE)/2
2030 IFSCORE=INT(SCORE),1THENX=0:REPEAT:I=I+1:UNTILSCORE=S(I):THETA=PI*I/10-PI
/20:IF(41>5):X=R*COS(THETA):Y=R*SIN(THETA):ELSEX=80:Y=330
2040 PROCRANDOM
2050 ENDPROC
2070 DEF PROCRANDOM
2080 D=RND(3)-2:E=RND(3)-2
2090 X=X+D*15:Y=Y+E*15
2100 ENDPROC
2120 DEF PROCWRITEND
2130 PRINTTAB(1,2)"                    "
2140 VDU5:GCOL3,5
2150 MOVE-640,504:PRINT"WINNER: ";
2160 PRINTNAME$(PLAYER)
2170 PRINT"ANOTHER GAME?"
2180 PRINT"Y/N"
2190 ENDPROC
2210 DEF PROCWRITENAME
2220 IFPLAYER=0THENPRINTTAB(32,2)"         "ELSEPRINTTAB(1,2)"
2230 SOUND1,-15,100,3
2240 PRINTTAB(1+33*PLAYER,2)NAME$(PLAYER)
2250 ENDPROC
2270 DEF PROCTHUD
2280 SOUND0,-15,6,5
2290 ENDPROC
2310 DATA6,13,4,18,1,20,5,12,9,14,11,8,16,7,19,3,17,2,15,10,6
```

(continued from page 173)

triangle 2 is drawn and then on the next pass of the loop two new points are calculated and two new triangles are filled in. In this way each segment of each sector is drawn using pairs of triangles.

Lines 1290 and 1300 print the scores — the array S — around the board by calculating the necessary points on the circle, and slightly adjusting the position for the numbers 18, 13, 10 and 15 so that the board appears balanced. Note that the Text and Graphics Cursor were combined in line 1190. Line 1360 draws the wires around the board using the Procedure Circle with the necessary radii being passed to it. The only thing left to do is draw the bull in the centre; again done by the Plot 85 command.

The game proceeds by a set of nested Repeat-until loops. The outer is controlled by the logical variable Finish which only becomes true when the player elects to end the game in lines 580, 590.

The next inner loop — lines 340-560 — is controlled by the logical variable Gameover which is set true in Procedure Totalise if the scoring is okay for a permissible finish to the game.

The next loop in is controlled by three variables, Dart, Gameover, and Bust. The latter is set true in Procedure Bust if the cumulative score — the variable TScore — for the three darts — controlled by the counter Dart — is greater than the score — Score (Player) — already obtained by the player.

In the innermost loop a series of Procedures first move the cross-hair around the board using the GCOL 3 command to first draw — line 980 — and then erase — line 1000 — a cross at a position X,Y which is randomly moved in a Repeat-Until loop unless the player moves it.

Within this loop a check is made — line 1050 — to see if the machine should play, in which case Procedure Mygame is invoked. Here a series of simple If statements is used to make a decision about where to land a dart. X and Y are set but to make it fairer X and Y are randomised by the same degree to which the cross-hair's movement is subjected.

Line 420 checks to see whether a wire has been hit. Because they were drawn in the background colour 2 — yellow — line 470 has to reset the logical variable Wireflag back to False if the point was really off the board or in the 25 region. This could have been allowed for in line 420. Note that the darts are drawn in yellow on a black background and so hitting a dart can result in a zero score and the message "wire". The rest of the program is concerned with drawing the dart, writing the score of each individual dart, and writing the scores of the players together with their names. The only complicated part occurs in Procedure Score. Here the position of the dart is first remembered by the arrays DartX and DartY for subsequent removal and then the radial distance of the dart from the board's centre is calculated in line 680. A series of Ifs then checks this radial distance R to provide initial scorings. The order of these is quite important because the number of checks can be kept to a minimum. Thus scores of 50, 25 and 0 are easily checked. After checking — line 710 — for the wire having been hit, the angle measured anti-clockwise round the board from the middle of the 3 o'clock sector — that scoring 6 — is calculated by the ACS function — line 720 — which may have to be adjusted if the dart is in the bottom half of the board.

The next line, 730, essentially takes this angle Theta and divides it by PI/10 to calculate which sector the dart is in and therefore which array element will provide the score. PI/20 is subtracted to allow for the counter-clockwise rotation around the circle.

Now here comes the slightly non-mathematical "adjustment" to make the scoring right. If the dart were to land in the bottom part of sector 6 the measured angle would be slightly negative and so 1 should be added to the calculation to give the array element. However, because of the way the sectors were drawn, the trigonometry is slightly awry and so I only add 0.7. This makes the scoring correct as it appears on the screen. Having found which sector the dart is in, it is an easy matter to check for doubles and trebles by looking at the radial distance from the centre of the board — lines 740-750. If the score is a double then the logical variable Double is set true. Note that it is also set true if 50 is scored because it is used to check for a legal end to the game in line 1560.

Synthesiser

Andrew Parker,
Warrington.

`CBM-64`

THIS PROGRAM uses all the features of the 64's sound chip, using the whole of the keyboard to play like a musical instrument. It occupies 10K of memory. The keyboard is set out as

```
5 POKE53280,0:POKE53281,0
10 PRINT"       "
15 PRINT"       "
20 PRINT"       THE COMMODORE 64 MUSIC SYNTHESISER    "
25 PRINT"       "
30 PRINT:PRINT"THIS USES THE FIRST VOICE OVER A RANGE"
40 PRINT"OF NEARLY 3 OCTAVES"
50 PRINT"THE WAVEFORM IS CHANGEABLE DURING USE BY"
53 PRINT"PRESSING :F1  FOR TRIANGLE, :F3  FOR SAWTOOTH"
56 PRINT"F5  FOR PULSE AND F7  FOR NOISE"
58 PRINT"ADSR AND PULSE LEVELS (IF REQUIRED)"
60 PRINT"ARE SET AFTER THESE INSTRUCTIONS"
70 PRINT"THE SHARPS ARE THE KEYS 2,3,4,6,7,9,0,+,-"
75 PRINT", AND :CLR HOME "
80 PRINT"THE NORMAL NOTES ARE FOR Z TO / AND"
90 PRINT"FROM 0 TO 1"
95 PRINT" :PRINT"        HIT ANY KEY TO START  "
96 GETA$:IFA$=""THEN96
98 PRINT"       "
100 DIMNO(64,2)
110 FORC=1TO62
120 READNO(C,1):NO(C,2)
130 NEXTC
140 DATA0,0,0,0,0,0,0,0,0,0,0,0,0
```
(listing continued from page 176)

(continued from page 175)

shown in the diagram, the keys from Z to the oblique stroke acting as normal notes and the keys S, D, G, H, J, L and , acting as the sharps. The same is true for the top two rows of keys. The sharps are on 2, 3, 4, 6, 7, 9, 0, +, £ and the CLR/HOME key. This gives a range of just over three octaves.

The waveform can either be triangle, sawtooth, pulse or noise. These govern the shape of the sound wave. The attack-decay and sustain-release determine how long it takes for a note to reach its peak volume — attack — to fall from its peak volume to sustain level — decay — how long it stays at this level — sustain — and how long the note takes to fall from the sustain level — release.

The waveform can be changed during the running of the program by using the function keys (see program listing). The instructions are contained in lines 10-95. Lines 100 to 230 read the high and low frequency values into an array. The place in the array into which the note is placed is governed by the value in the keyboard buffer (location 197) for that key. The main locations are:

```
54296-Volume control
54273-High frequency
54272-Low frequency
54276-Waveform
54275-High pulse
54274-Low pulse
54277-Attack/Decay
54278-Sustain/Release
```

These locations are used for voice one only, voices 2 and 3 have different locations. In Line 462, Poke 198,0 clears the keyboard buffer after each key.

(listing continued from page 175)

```
150 DATA27,56,25,177,0,0,30,141,8,147,9,21,28,214
160 DATA0,0,0,34,75,9,159,40,200,38,126,14,32
170 DATA0,0,0,34,75,9,159,40,200,38,126,14,32
180 DATA0,0,12,216,15,156,43,52,11,114,48,127
190 DATA45,138,15,70,54,111,16,47,0,0,51,97
200 DATA14,107,61,126,57,172,18,42,0,0
210 DATA19,63,20,100,64,88,17,37,72,169,68,149
220 DATA0,0,81,161,0,0,0,0,76,252,21,154
230 DATA0,0,0,0,0,0,24,63,0,0,0,0,22,227,0,0
240 VOL=54296
250 HFR=54273
260 LFR=54272
270 WFM=54276
280 HPE=54275
290 LPE=54274
300 AD=54277
310 SR=54278
320 PRINT"WAVEFORM "
330 PRINT"17 = TRIANGLE"
340 PRINT"33 = SAWTOOTH"
350 PRINT"65 = PULSE"
360 PRINT"129= NOISE"
365 INPUTR
370 INPUT"ATTACK/DECAY";L(1)
380 INPUT"SUSTAIN/RELEASE";L(2)
390 IFR<>65THEN420
400 INPUT"HIGH PULSE  (0-15)";P(1)
410 INPUT"LOW PULSE  (0-255)";P(2)
420 REM**KEYBOARD SCAN**
430 FOKEVOL,15
435 POKELPE,P(2):POKEHPE,P(1)
440 POKEAD,L(1)
450 POKESR,L(2)
455 POKEHFR,R
460 S=PEEK(197)
462 POKE198,0
463 GOTO82000
465 IFS=60THENPOKEVOL,0:POKEWFM,0:POKEAD,0:POKESR,0
480 IFS=64THEN460
500 POKEHFR,NO(S,1)
510 POKELFR,NO(S,2)
515 IFPEEK(197)=STHEN515
530 POKEWFM,0
540 GOTO430
500 POKEVOL,0
610 GOTO430
2000 IFS=4THENR=17
2010 IFS=5THENR=33
2020 IFS=6THENR=65:GOTO430
2030 IFS=3THENR=129
2040 RETURN
5200 POKEAD,0:POKESR,0:POKEWFM,0
9999 PRINT"J"
```

Tape examiner

M Salmon,
Abingden,
Oxfordshire.

SPECTRUM

THE SPECTRUM saves files as two blocks — header and data. The data may be of any length and could represent anything, so the header is needed to tell the Spectrum how to interpret the data. The header is composed of 17 bytes, as follows:

offset	description
0	file type: 0=program, 1=number array, 2=string array, 3=bytes
1-10	filename in ASCII
11-12	total length in bytes of data to be read in
13-14	special
15-16	program length in bytes

Bytes 13 and 14 are used for any specific information required. For type 0 files it contains the auto-run line number — unless it is more than 32767 which signifies load only. For type 3 files it contains the start address for Load "" Code with no parameter. For types 1 and 2 only byte 14 is used. This contains packed information for the aid of the floating point calculator. Bits 0-4 give the ASCII code minus 64 of the original array saved and bit 6 tells what type of array it is (0=number, 1=string). Bit 7 is always set.

The table above is given with offsets as the method used for reading files from tape is by calling a routine with a buffer pointed to by the 16 bit index register IX.

The program contins a small machine routine to do this:

175	XOR A
55	SCF
221,33,16,127	LD IX,32528
205,86,5	CALL 1414
201	RET

This is set up by lines 10-30. Line 40 sets B to point to the buffer and sets up a user-defined function to extract a double-byte number from an offset X.

Line 50 calls the machine-code subroutine and the Spectrum waits for a header to come in from the cassette port.

```
10 CLEAR 32511
20 FOR a=32512 TO 32521: READ
   b: POKE a,b: NEXT a
30 DATA 175,55,221,33,16,127,2
   05,86,5,201
40 LET b=32528: DEF FN a(x) =PE
   EK (b+x)*256+PEEK (b+x+1)
50 RANDOMIZE USR 32512
60 LET c=PEEK b
70 IF c>3 THEN GO TO 50
80 PRINT "Filename: "
90 FOR a=b+1 TO b+10: PRINT CH
   R$ PEEK a;: NEXT a
100 PRINT : PRINT TAB 4;"Type:
   "
110 GO SUB 1000+100*c
120 PRINT : PRINT
125 POKE b,255
130 GO TO 50
1000 PRINT "Program"
1010 PRINT "Total length: ";FN a
   (11);" bytes"
1020 PRINT "Program length: ";FN
   a(15);" bytes"
1030 IF FN a(13)>9999 THEN PRINT
   "Load only": RETURN
1040 PRINT "Runs from line ";FN
   a(13)
1050 RETURN
1100 PRINT "number array"
1110 LET a$="": GO TO 1220
1200 PRINT "character array"
1210 LET a$="$"
1220 PRINT "Array length: ";FN a
   (11);" bytes"
1230 LET d=PEEK (b+14)
1240 PRINT "Original array name:
    ";CHR$ (64+32*(d/32-INT (d/32))
   )a$
1250 RETURN
1300 IF FN a(11)=6912 AND FN a(1
   3)=16384 THEN PRINT "screen imag
   e": RETURN
1310 PRINT "bytes"
1320 PRINT "Start address: ";FN
   a(13)
1330 PRINT "Length: ";FN a(11);"
    bytes"
1340 RETURN
```

Line 60 extracts the filetype and line 70 skips the output section if it decides that this is not a header. Lines 80-100 print the filename and line 110 chooses the appropriate routine to use for decoding on the basis of the filetype. This means that it is vitally important that the routines starting at lines 1000, 1100, 1200 and 1300 remain at these lines — otherwise numbering is unimportant. Line 1300 is the only other interesting point — it automatically decides whether a bytes file is a screen image or not.

Spectrum owners with an understanding of machine-code and a good disassembler should find the area of the ROM between 4C2 and 9F3 — hex — interesting to inspect as the tape-handling routines are self-contained. I would advise starting at 605 hexadecimal as this is the highest level of tape handling. The prospective decoder should be helped by the knowledge that the low byte of TADDR will contain a byte ranging from E0-E3 corresponding to the keywords Save, Load, Verify and Merge respectively.

Memory man

TR Carey,
Southampton,
Hampshire.

ZX-81

IF YOU want to know how much memory has been used by various things this program will run on any ZX-81 but it is not really suitable for the 1K machine as the memory is very limited. Using the memory map on page 177 of the ZX-81 manual and the systems variables on pages 177/9. The systems variables are used to find the addresses of different parts of the memory. This program is very useful when writing programs and it can be removed afterwards.

```
9996 PRINT TAB 8;"ZX 81 MEMORY",
,,"TOTAL RAM",((PEEK-16388+256*P
EEK 16389)-16384)/1024;" K"
9997 PRINT ,,"BASIC PROGRAM",(PE
EK 16396+256*PEEK 16397)-16509;"
BYTES"
9998 PRINT ,,"VARIABLES",(PEEK 1
6404+256*PEEK 16405)-(PEEK 16400
+256*PEEK 16401)-1;" BYTES"
9999 PRINT,, "SPARE MEMORY",(PEE
K 16388+256*PEEK 16389)-(PEEK 16
404+256*16405);" BYTES"
```

Definer

Brian Etherington,
Hanslope,
Milton Keynes.

BBC

THIS IS A MULTICOLOUR character definer for a 32K BBC Micro. The cursor is moved around an eight by eight grid to define a new character. As new pixels are added or deleted from the new character, the VDU statement parameters are updated on the right-hand side of the grid, in line with each of the eight grid rows. Similarly, the character is displayed in mode 2 graphics to the left of the larger grid. At the bottom of the screen in a 16-colour paint palette, the current ink colour is indicated by the arrow. The ink colour is changed by pressing the space bar. In this

way, overlaid multicoloured graphics characters may be defined and automatically displayed. The VDU parameters are displayed for each colour as you step through the paint palette.

The program has been written in fully-structured Basic and makes extensive use of BBC Basic procedures. The only Goto statement in the entire program is at line 140. Note the procedure Cursoron at line 1200 redefines a different form of cursor to that which is normally available. I believe that this program represents a valuable software-development tool for the BBC microcomputer, and is unusual in that it allows for multi-coloured

character generation. In addition, the program is an example of good programming technique for those just learning the craft. Multicoloured characters can be printed using the VDU5 and move commands. The following example prints a character in red, white and blue. It assumes that characters 224-226 have been defined by the user:

```
 5 MODE 2
10 BLUE$ = CHR$ 18 + CHR$ 0 + CHR$ 4
20 RED$ = CHR$ 18 + CHR$ 0 + CHR$ 1
30 WHITE$ = CHR$ 18 + CHR$ 0 + CHR$ 7
40 A$ = BLUE$ + CHR$ 224 + CHR$ 8 +
   RED$ + CHR$ 225 + CHR$ 8 + WHITE$ +
   CHR$ 226
50 VDU 5
60 MOVE 640, 512 : PRINT A$
```


CHARACTER DEFINER

BLUE

Use normal cursor movement keys to position cursor.

Make a mark using the COPY key.

Erase a mark using the DELETE key.

Press SPACE BAR to change ink colour.

Press ESCAPE to define new character

```
10 REM ***************************************
20 REM *                                     *
30 REM * MULTI-COLOUR CHARACTER DEFINITION PROGRAM *
40 REM *                                     *
50 REM * AUTHOR :- B.J.W.ETHERINGTON         *
60 REM *              JANUARY 1983           *
70 REM *                                     *
80 REM ***************************************
90 REM
100 ON ERROR RUN
110 DIM GRID%(8,8),ROW%(15,9),SHADE$(15)
120 MODE 2
130 PROC_INITIALISE
140 PROC_INPUT:GOTO 140
150 DATA RED,GREEN,YELLOW,BLUE,MAGENTA,CYAN,WHITE,FLASH BLACK-WHITE,FLASH RED-
CYAN,FLASH GREEN-MAGENTA,FLASH YELLOW-BLUE,FLASH BLUE-YELLOW,FLASH MAGENTA-GREEN
,FLASH CYAN-RED,FLASH WHITE-BLACK
160 REM **************************************** INITIALISE
170 DEF PROC_INITIALISE
180#FX4,1
190FOR I%=1 TO 8:FOR J%=1 TO 8:GRID%(I%,J%)=0:NEXT:NEXT
200FORI%=1 TO 15:ROW%(I%,9)=0:READ SHADE$(I%):NEXT
210X%=1:Y%=1
220 VDU 23,224,&FF,&FF,&FF,&FF,&FF,&FF,&FF,&FF
230 VDU 23,240,3,1,1,1,1,1,1,3
240 VDU 23,241,0,0,0,0,0,0,129,255
250 VDU 23,242,255,129,0,0,0,0,0,0
260 VDU 23,243,192,128,128,128,128,128,128,192
270 VDU 23,244,16,16,16,16,84,56,16,0
280PROC_SCREEN
290PROC_CURSORON
300PROC_MOVE(0,0)
310PROC_DISPLAY
320ENDPROC
330REM **************************************** SCREEN
340DEF PROC_SCREEN
350CLS:COLOUR 7
360PRINT " CHARACTER DEFINER"
370 COLOUR 2
380 PRINT TAB(0,14)"Use normal cursor";
390PRINT TAB(0,15)"movement keys to";
400PRINT TAB(0,16)"position cursor.";
410 PRINT TAB(0,18)"Make a mark using";
420 PRINT TAB(0,19)"the COPY key.";
430 PRINT TAB(0,21)"Erase a mark using";
440 PRINT TAB(0,22)"the DELETE key.";
450 PRINT TAB(0,24)"Press SPACE BAR to";
460 PRINT TAB(0,25)"change ink colour.";
470 PRINT TAB(0,27)"Press ESCAPE to";
480 PRINT TAB(0,28)"define new character";
490 FOR I%=1 TO 15:COLOUR I%:PRINT TAB(I%+1,31);CHR$(224):NEXT
500 INK%=6
510 PROC_INK
520PRINT TAB(6,1)STRING$(8,CHR$(241))
530PRINT TAB(6,10)STRING$(8,CHR$(242))
540 VDU 31,5,2,240,8,10,240,8,10,240,8,10,240,8,10,240,8,10,240,8,10,240,8,10,
240
550 VDU 31,14,2,243,8,10,243,8,10,243,8,10,243,8,10,243,8,10,243,8,10,243,8,10
,243
560 ENDPROC
570REM **************************************** INPUT
580DEF PROC_INPUT
590 A=GET
600 IF A=&88 PROC_MOVE(-1,0):ENDPROC
610 IF A=&89 PROC_MOVE(1,0):ENDPROC
620 IF A=&8B PROC_MOVE(0,-1):ENDPROC
630 IF A=&8A PROC_MOVE(0,1):ENDPROC
640 IF A=&20 PROC_INK:ENDPROC
650 IF A=&87 PROC_MARK(INK%):ENDPROC
660 IF A=&7F PROC_MARK(0):ENDPROC
670ENDPROC
680REM **************************************** MOVE
690DEF PROC_MOVE(XINC%,YINC%)
700X%=X%+XINC%:Y%=Y%+YINC%
710IF X%>8 X%=1
720IF X%<1 X%=8
730IF Y%>8 Y%=1
740IF Y%<1 Y%=8
750 PRINT TAB(X%+5,Y%+1);
760ENDPROC
770REM **************************************** MARK
780DEF PROC_MARK(A%)
790IF GRID%(Y%,X%)=A% ENDPROC
800COLOUR A%
810VDU 224,8
820IF GRID%(Y%,X%)<>0 TINK%=GRID%(Y%,X%):GRID%(Y%,X%)=0:PROC_CALC(TINK%)
830GRID%(Y%,X%)=A%
840 PROC_DISPLAY
850ENDPROC
860REM **************************************** INK
870DEF PROC_INK
880PROC_CURSOROFF
890COLOUR 7
900PRINT TAB(INK%+1,30)" ";
910INK%=INK%+1
920IF INK%>15 INK%=1
930PRINT TAB(INK%+1,30);CHR$(244);
940PRINT TAB(0,11);STRING$(40," ");
950 PRINT TAB(0,11);SHADE$(INK%);
960COLOUR INK%
970PROC_VDU
980PROC_MOVE(0,0)
990PROC_CURSORON
1000ENDPROC
1010REM **************************************** DISPLAY
1020DEF PROC_DISPLAY
1030PROC_CURSOROFF
1040PROC_VDU
1050 VDU 5
1060 MOVE 192,847
1070 VDU 9,127
1080FOR I%=1 TO 15
1090IF ROW%(I%,9)=1 PROC_SHOW(I%)
1100NEXT
1110 VDU 4
1120 COLOUR INK%
1130PROC_MOVE(0,0)
1140PROC_CURSORON
1150ENDPROC
1160 REM **************************************** CURSOROFF
1170 DEF PROC_CURSOROFF
1180VDU 23,0,10,32,0,0,0
1190 ENDPROC
1200 REM **************************************** CURSORON
1210 DEF PROC_CURSORON
1220VDU 23,0,10,67,0,0,0
1230VDU 23,0,11,4,0,0,0
1240ENDPROC
1250 REM **************************************** SHOW
1260 DEF PROC_SHOW(A%)
1270 GCOL 0,A%
1280VDU 23,A%+224,ROW%(A%,1),ROW%(A%,2),ROW%(A%,3),ROW%(A%,4),ROW%(A%,5),ROW%(A
%,6),ROW%(A%,7),ROW%(A%,8)
1290 MOVE 192,847
1300 VDU A%+224
1310 ENDPROC
1320 REM **************************************** CALC
1330 DEF PROC_CALC(A%)
1340ROW%(A%,9)=0
1350FOR I%=1 TO 8
1360ROW%(A%,I%)=0
1370FOR J%=1 TO 8
1380IF GRID%(I%,J%)=A% ROW%(A%,I%)=ROW%(A%,I%)+(2^(8-J%)):ROW%(A%,9)=1
1390NEXT:NEXT
1400ENDPROC
1410 REM **************************************** VDU
1420DEF PROC_VDU
1430PROC_CALC(INK%)
1440COLOUR 7
1450FOR I%=1 TO 8
1460PRINT TAB(15,I%+1)"   ";
1470PRINT TAB(15,I%+1);ROW%(INK%,I%)
1480NEXT
1490ENDPROC
```

Rally racer

D T C Breslin,
Byfleet,
Surrey.

ATOM

HERE IS A program for the Acorn Atom which

tests a player's ability to steer down a twisting lane. Using a combination of Atom Basic and resident assembler, the program uses under 2K of memory, and also graphic mode 2a — colour — so the screen RAM and floating point ROM must be fitted. The road is shown in a 3-D image as white dots and every time

```
10 IF ?#99<>0 W=0;?#99=0
20 GOS.s
30 @=0
40 DIM LL5
50 F.J=0TO20;LL(J)=-1;N.
60 P.$21;F.J=1TO2;P=#2800
```

(listing continued from page 179)

```
70C
 80:LL0 JSR LL3;LDA #9A;CLC;ADC #9B;STA #9B;STA #3B1F;STA#3A1F
 90 JSR #FE71;LDA 00;STA #90
100 CPY @1;BNE LL1
110 LDA #9E;STA #90;CLC;ADC #9B;STA #9B
120 LL1 CPY @3;BNE LL2
130 LDA #9F;STA #90;CLC;ADC #9B;STA #9B
140 LL2 LDA #B002;EOR@4;STA#B002;RTS
150 LL3 LDX 00;STX #5B;STX #5D;STX #5E
160 LL4 STX #5C
170 LDA #3A00,X;STA #5A;CLC;ADC #90;SEC;SBC @1;STA #39FF,X
180 JSR LL5;LDX #5C;LDA #39FF,X;STA #5A;INC #5E
190 DEC #5C;JSR LL5;DEC #5E;INC #5C
200 LDX #5C;LDA #3B00,X;STA #5A;ADC #90;STA #3AFF,X
210 JSR LL5;LDX #5C;LDA #3AFF,X;STA #5A;INC #5E
220 DEC #5C;JSR LL5;DEC #5E
230 LDX #5C;INX;INX;CPX @32;BNE LL4
240 LDA #87CE;ORA #87CF;ORA #87D0;ORA #87D1;STA #91;RTS
250:LL5 JMP(#3FE)
260J;N.;P.$6
270 C=0;L=0;B=#B002;H=2;K=1;?#9E=1;?#9F=254
280hCLEAR2;COLOUR1;?#9B=64;!#87EE=-1;GOS.d
290 F.J=0TO32;J!#3A00=32+J;J!#3B00=96-J;N.
300 LI.LL3;F.J=0TO180;WAIT;N.
```

```
310aK=R.%H;?#9A=K;F.J=0TOA.R.%5+3;C=C+1
320 IF C%500=0;H=H+1;?#9E=H-1;?#9F=-H+1;GOS.d
330 LI.LL0
340 ?B=?B:4
350 IF ?#91<>0;?#15=?#15-1;G.9
360 ?B=?B:4;N.;G.a
370bP.$12"YOU SCORED "C" POINTS"''''
380 IF C>W;W=C
390 P." HIGH SCORE IS "W"''''
400 P."PRESS [SPACE BAR] FOR ANOTHER GO";LI.#FFE3;RUN
410bF.J=0TO200;?B=?B:4&R.;N.;L=L+1
420 IF L=3;G.b
430 G.h
440dP.$7;IF H=2;G.460
450 F.O=1TO H;MOVE((O-2)*2),61;DRAW((O-2)*2),63;N.
460 R.
470CLEAR0;P.$30
480 F.J=35TO45;MOVE22,J;DRAW45,40;N.
490 P.'''       "$9"lombard"''''
500 P."STEER YOUR CAR AS FAR AS YOU CANDOWN THE ROAD BY THE"
510 P." KEYS :-"''
520 P." [ "$95" LEFT"''
530 P." ] "$95" RIGHT"''
540 P.''"     PRESS SPACE BAR TO CONTINUE"
550 LINK#FFE3
560 R.
```

(continued from page 179)

500 of these disappear off the bottom of the screen, the degree of difficulty increases until the player crashes three times. The score obtained and high score is printed and the game starts again.

Extra tools

Peter Barney,
Letchworth,
Hertfordshire.

SPECTRUM

USERS OF Bobby Rao's toolkit machine-code routines for the Spectrum — page 77, February, *Your Computer* — may find these routines useful also. List 3 and list 4 are high-resolution left and right scrolls, but in each case, the byte listed as 55 or 63 — use only one — will scroll with wrap-around if 63, or with a white pixel from left or right, as appropriate, if 55. This same byte Poked 0 gives inverse characters with wrap around.

List 1 and list 2 are high-res up or down scrolls, which may be of interest as they are much shorter — 53 and 55 bytes — than the 99 and 97 of the published routines.

I have given the listings in decimal as I find these far easier to enter than hexadecimal. A suitable routine for entering the codes is given on page 180 of the manual.

The trap

Colin Hagreen,
Orpington,
Kent.

DRAGON

TRAP IS A game for two players, each player directing a line around the screen trying to force their opponent to hit one of the tracks

```
   5 REM Demo Program
  10 CLEAR (PEEK 23675+256*PEEK 23676-200)
  60 GO SUB 120
  65 IF INKEY$="l" THEN POKE 32533,0: POKE 32563,0
  70 IF INKEY$="u" THEN RANDOMIZE USR 32400
  75 IF INKEY$="w" THEN POKE 32533,63: POKE 32563,63
  80 IF INKEY$="d" THEN RANDOMIZE USR 32460
  85 IF INKEY$="s" THEN GO SUB 120
  90 IF INKEY$="l" THEN RANDOMIZE USR 32520
  95 IF INKEY$="x" THEN POKE 32533,55: POKE 32563,55
 100 IF INKEY$="r" THEN RANDOMIZE USR 32550
 105 GO TO 65
 110 GO TO 150
 120 FOR i=0 TO 21: FOR j=0 TO 31: PRINT AT i,j;CHR$ (133+j): NEXT j: NEXT i
 125 PRINT AT 10,5;"Type u for up       ";AT 11,5;"    d for down     ";A
T 12,5;"   r for right      ";AT 13,5;"    l for left      ";AT 14,5;" and
  s for start again  ";AT 15,5;"    w for wrap around";AT 16,5;" i for inverse ";
CHR$ 34;"   ";CHR$ 34;"   ";AT 17,5;"      x cancels ";CHR$ 34;"   ";CHR$ 34;"
 130 RETURN
```

LIST 4

32520	33	32521	255	32522	87	32523	6	32524	192
32525	197	32526	229	32527	221	32528	225	32529	221
32530	126	32531	225	32532	7	32533	55	32534	63
32535	6	32536	32	32537	203	32538	22	32539	43
32540	16	32541	251	32542	193	32543	16	32544	236
32545	201								

LIST 3

32550	33	32551	0	32552	64	32553	6	32554	192
32555	197	32556	229	32557	221	32558	225	32559	221
32560	126	32561	31	32562	15	32563	55	32564	63
32565	6	32566	32	32567	203	32568	30	32569	35
32570	16	32571	251	32572	193	32573	16	32574	236
32575	201								

LIST 2

32400	33	32401	31	32402	64	32403	84	32404	93
32405	37	32406	6	32407	3	32408	197	32409	6
32410	8	32411	197	32412	6	32413	8	32414	197
32415	36	32416	1	32417	32	32418	0	32419	237
32420	184	32421	17	32422	32	32423	0	32424	25
32425	84	32426	93	32427	193	32428	16	32429	240
32430	1	32431	32	32432	248	32433	9	32434	193
32435	16	32436	230	32437	62	32438	7	32439	132
32440	103	32441	193	32442	16	32443	220	32444	33
32445	223	32446	87	32447	6	32448	32	32449	35
32450	54	32451	0	32452	16	32453	251	32454	201

LIST 1

32460	33	32461	255	32462	87	32463	84	32464	93
32465	36	32466	6	32467	3	32468	197	32469	6
32470	8	32471	197	32472	6	32473	8	32474	197
32475	37	32476	1	32477	32	32478	0	32479	237
32480	184	32481	17	32482	32	32483	0	32484	25
32485	84	32486	93	32487	193	32488	16	32489	240
32490	1	32491	224	32492	7	32493	9	32494	193
32495	16	32496	230	32497	62	32498	249	32499	132
32500	103	32501	193	32502	16	32503	220	32504	37
32505	6	32506	32	32507	35	32508	54	32509	0
32510	16	32511	251	32512	201					

formed or the boundaries of the court. Beware though as you can sometimes escape by cutting through a diagonal line. Be careful not to centre the joystick or you lose the game.

To start a game both players have to press the fire buttons. The overall winner is the first to score 10 wins. The lower-case letters in lines 350 and 360 are typed as inverse characters. PCopy is used to store the courts in another area of memory and lines 120 and 130 bring them on to the display faster than could be obtained by drawing them each time.

```
>   5 REM  (C) C. HAGREEN '83
   10 PCLEAR6
   20 PMODE1,3
   30 PCLS
   40 LINE(0,0)-(0,192),PSET:LINE-(255,192),PSET:LINE-(255,0),PS
ET:LINE-(0,0),PSET
   50 PMODE1,5
   60 PCLS
   70 PCOPY3 TO 5:PCOPY4 TO 6
   80 LINE(50,50)-(50,150),PSET:LINE(200,50)-(200,150),PSET
   90 AS=0:CS=0
  100 GOTO 280
  110 X=RND(2)
  120 PCOPY(X*2)+1 TO 1
  130 PCOPY(X*2)+2 TO 2
  140 PMODE1,1:SCREEN1,0
  150 X=120:Y=90:X1=100:Y1=90
  160 X=X+2*(JOYSTK(0)>51)-2*(JOYSTK(0)<12)
  170 Y=Y+2*(JOYSTK(1)>51)-2*(JOYSTK(1)<12)
  180 IF PPOINT(X,Y)<>1 THEN 270
  190 PSET(X,Y,2)
```

```
  200 X1=X1+2*(JOYSTK(2)>51)-2*(JOYSTK(2)<12)
  210 Y1=Y1+2*(JOYSTK(3)>51)-2*(JOYSTK(3)<12)
  220 IF PPOINT(X1,Y1)<>1 THEN 250
  230 PSET(X1,Y1,3)
  240 GOTO160
  250 CS=CS+1
  260 GOTO280
  270 AS=AS+1
  280 CLS:PRINT"          T R A P":PRINT"          ----------"
  290 PRINT@128,"LEFT PLAYER          RIGHT PLAYER"
  300 PRINT@196,AS:PRINT@216,CS
  310 IF AS=10 OR CS=10 THEN 350
  320 PRINT"PRESS BOTH FIRE BUTTONS"
  330 IF PEEK(65280)<>124 AND PEEK(65280)<>252 THEN 330
  340 GOTO110
  350 IF AS=10 THEN PRINT:PRINT" left player wins" ELSE PRINT:PR
INT" right player wins"
  360 PRINT:PRINT:PRINT"   PRESS enter"
  370 A$=INKEY$:IF A$<>CHR$(13) THEN 370
  380 RUN
```

A £15 book token will be awarded to the first correct solution drawn from the competition bag. All entries must be at the *Your Computer* offices by the last working day in May. The name of the winner, the solution, and a competition report will be published in the July, 1983 issue of *Your Computer*.

If you want to set a competition for Competition Corner, remember that the simplest solution should be calculable by a short program rather than by any other form of reckoning.

THE HALL OF THE TROLL

HERE IS A map of the trolls' burrow — and you are out to make your fortune. How many coins should you pick up in the first chamber you enter to survive the trip with the most diamonds? As usual, all the passageways are one-way, as shown.

BY ANTHONY ROBERTS

Troll gives you as many coins as you have diamonds

A troll swaps your diamonds for an equal number of beads . . . then gives you as many diamonds as you have coins

Troll takes one bead from you —then sends you up if you've any beads left: Else sends you down

Troll gives you an emerald and as many diamonds as you have coins

Troll takes one emerald and sends you up . . . but if you haven't any, send you right

Troll sells you as many apples as you have money to buy at ten coins per apple

Troll buys all your apples at 1 coin for each

Troll takes all your coins and returns them to the store: Takes all your diamonds and throws them outside: He then sends you left if you've any apples with you. If not, he'll let you go (and of course you can pick up those diamonds he's thrown away)—after counting the coins in the store. If there aren't exactly 1000, he'll suspect foul play, and probably kill you . . .

Take some— not all—of the 1000 coins in the store of coins here

COIN STORE

Competition results

IN MARCH'S competition to win a Lynx computer competitors were asked to complete the sentence, "A Lynx would bring out the animal in me because . . .".

The prospect of winning a Lynx may not have brought out the animal in everyone but it certainly brought out the punsters, in force. "I could relion my feline instincts for purrfect programming", wrote G Bockhurst, while H Howarth suggested "It's the big-byte cat-grrrreat grrraphics and a purrfect purrocessor to boot".

The winning entry came from the appropriately-named Dave Bull, 86 Milton Road, Southampton, Berkshire, who wrote "It's the purrfect way to be an on-line feline".

Some of the other puns were more obscure. We are still trying to puzzle out A Ridley's "If ewe bison, ewe kangaroo terrier-bull byte otter it".

More straightforward were M Glass's "It's ears ahead", and from A Patrick, "It provides the missing links to make me a cool Computer cat".

Mrs Allemand struck a suitably aggressive note, worthy of Marvel comics, with "as Catwoman I'd have power over those ZX-81 jokers". E Jupp's entry, however, took a more relaxed line and confided owlishly "I wynx and blynx, but the Lynx thynx".

Our problem setter slipped up with the Telepathic Dangers competition by setting a problem which had not one but two solutions. Two pairs of segments can be discarded from the cube of On'ey: C and E, and B and F. Many of the entries spotted the blunder and gave both correct solutions.

We awarded the £15 book token to P Hoskins, 30 Springfield Park, Holyport, Berkshire.

He enclosed a program for the Spectrum which not only worked out the solutions but plotted the diagrams of the two possible cubes.

New! Unique! Exclusive!

A POWERFUL MICRO SYSTEM – PART 1

Our microcomputer system is built round the 6809 microprocessor. It's the most powerful 8-bit available, uses the fast, efficient Forth language, and provides rapid data storage and retrieval on a floppy disc. Don't miss the exclusive construction series beginning this month. See also a design for an inexpensive but sophisticated digital voltmeter that measures up to 8 different sources. Data can either be read direct by, or displayed on, a microcomputer monitor.

THERE'S NOT ONLY RADIO COMMUNICATIONS BUT MUCH MORE BESIDES IN WIRELESS WORLD – IN FACT, EVERYTHING IN ELECTRONICS. FOR THE CURRENT STATE OF THE ART IN ELECTRONICS, WIRELESS WORLD IS ESSENTIAL READING.

FOR EVERYTHING IN ELECTRONICS
Wireless world
May issue out now 80p

EXPLORING SPECTRUM BASIC
Complements the Sinclair manual by explanations of BASIC programming techniques supported by over 50 full games & serious programs. 191 pages. £4.95

THE EXPLORERS GUIDE TO THE ZX81
The book for the ZX81 enthusiast. 152 pages of games, application and utility programs plus much useful information on machine language and hardware. "Immediate and lasting value" (P.C.W.) £4.95

Available from leading bookshops and computer stores, or direct from Timedata (U.K. P&P free, overseas customers add £1.50 per item for surface mail).

TIMEDATA Ltd. Dept. A
16 Hemmells, Laindon, Basildon, Essex. SS15 6ED
Tel.: (0268) 418121

TIMEDATA

MORE MEMORY FOR YOUR JUPITER ACE

PACER 16K EXPANDABLE RAM
The uniquely expandable 16K Ram Pack, similar in concept to the 'ZX Panda' but for the incredible Jupiter Ace. Complete in attractive, solidly built, injection moulded case. For more power to faster FORTH . . . you need a Pacer for **ONLY £29·95**

16K EXPANSION MODULE
Increase your Pacer potential to 32K with the plug-in 16K Expansion Module for **ONLY £19·95**

AFDEC
ELECTRONICS LTD
318 Kempshott Lane, Basingstoke Hants RG22 5LT

Quantity	Item	Amount
	Pacer 16K Expandable Ram @ £29.95 inc.	
	Pacer 16K Expansion Module @ £19.95 inc.	
	TOTAL	

I wish to order the following

All prices include VAT and post and packaging. All items include a full year guarantee. Products are stocked and usually despatched within 10 days. Overseas customers add £4 post and packing. Cheques/Postal Orders etc. should be made payable to AFDEC Electronics Ltd.

Card No
Signature
Name
Address

At last.
A range of software that's as well designed as the Dragon 32.

Six months ago, most computer users thought a Dragon was something out of an adventure game.

How times change. Nowadays, it's almost impossible to pick up a computer magazine without reading some new tribute to the Dragon 32.

THE FIRST FAMILY COMPUTER

We called it the first family computer. And we believe that much of its success derives from the fact that it has been superbly designed for the job it has to do.

Design that incorporates 32K RAM of memory at a real value-for-money price level.

Design that utilises the advanced 6809E microprocessor.

Plus a range of features that help make the world of computers accessible to first-time users. Like colour drawings. 5 octaves of music.

And plenty of power and versatility to keep up the interest as your experience increases.

SOFTWARE TO MATCH
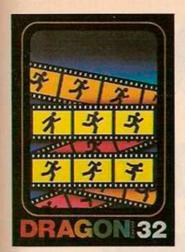
Designing a range of software that does justice to the Dragon was a task we approached with as much care as designing the computer itself.

The results are now available – games, educational programs, hints on programming – conceived by some of the best software houses in the world, to help users get even more out of their machines.
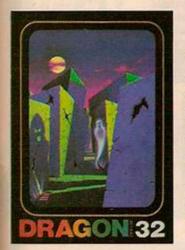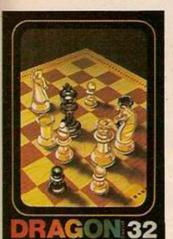
Graphic Animator. A cassette that makes use of the Dragon's ability to draw on the screen. It shows you how to create simple cartoons – then bring them to life.

Personal Finance. This cassette allows you to keep track of all the family's finances, from bills to bank statements.
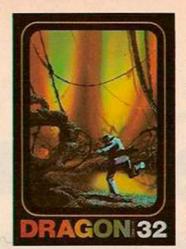
Ghost Attack. On cartridge. You're in a maze, and you must find your way out. But you're not alone – you're pursued by ghosts!

Chess. A great partner to play against again and again – with one major drawback. It's very hard to beat. Test yourself against nine different levels. On cartridge.

Typing Tutor. Only a keyboard as sophisticated as the Dragon's could allow you a program like this. You use the Dragon's own keyboard to learn to type. On cassette.

Cave Hunter. Finding the hidden gold is hard. But surviving the attacks of malevolent creatures may well prove to be impossible. Cave Hunter is on cartridge.

Dragon Selection. A cassette of utility programs, allowing you to create your own database. Write your own tunes – or learn other language vocabularies.

Starship Chameleon. A cartridge game. You find yourself under attack by the dreaded Gabolators! Their task is to take your planet – yours is to protect it.
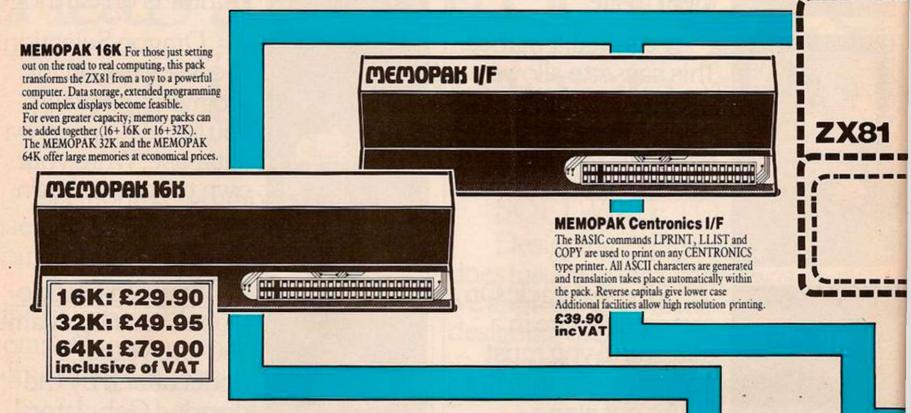
Computavoice. This cassette actually enables your Dragon to speak to you.

These are just some of our titles, and every month, we'll be adding more and more.

What you get out of a computer depends on what you put in. And with a Dragon 32 and Dragon software, there's enough challenge and satisfaction for anyone. But then, that's what it's designed for.

DRAGON 32
The first family computer.

PROGRAM DIRECT

DRAGON
SOFTWARE

SUBMARINE COMMAND

Take a modern Nuclear Submarine on anti-shipping patrol across the Atlantic. Stalk the enemy's convoys and sink his ship — but beware — enemy warships will be hunting you!.
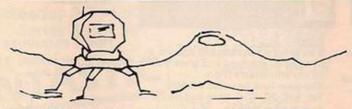
Great new game especially written for **Dragon** with full colour graphics in Machine Code.   **£7.95 (inc.)**

FEATURES INCLUDE:

Animated map and sonar screen
View the enemy through the periscope
Conventional and homing torpedoes
Surface—surface missiles
The enemy uses surface warships and aircrafts against you
Full status report
Full easy to understand instruction sheet.

DRAGON GAMES PACK
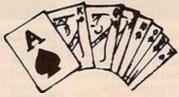
LUNAR LANDER o BOMBER o BREAKOUT — All in one cassette.

Exciting and addictive.   **£5.95 (inc.)**

MASTER CARDS PLAYER

Sophisticated implementations of two of the most popular card games **POKER** and **PONTOON**.

Each program uses high resolution colour graphics. Bluff or be bluffed.   **£6.95 (inc.)**

GREAT NEW GAMES FOR YOUR DRAGON 32 COMPUTER

OUR GUARANTEE

If our product ever fails to load, simply return it within 14 days and we will immediately replace it with refund for the postage.

All prices include VAT, Postage & Packing.

Send Cheques / P.O. made payable to:

**PROGRAM DIRECT LIMITED
37B NEW CAVENDISH STREET
LONDON W1M 8JR**

DEALERS ENQUIRIES WELCOME

Please send me the following software immediately:

☐ **Submarine Command**   £7.95 inc.
☐ **Master Cards Player**   £6.95 inc.
☐ **Dragon Games Pack**   £5.95 inc.

I enclose a Cheque / P.O. for £ . . . .

NAME: . . . .

ADDRESS: . . . .

. . . .

. . . . Postcode: . . . .

UNEXPANDED VIC 20

VIC 20 SOFTWARE FROM INTERCEPTOR MICRO'S

UNEXPANDED VIC 20

INSTANT IMAGE TRANSFER TO ZX SPECTRUM RD DIGITAL TRACER £55.50

Please send me ....RD DIGITAL TRACERS at £55.50 each (including P&P and VAT). I enclose cheque for £ .... payable to RD Laboratories Ltd.

Name ....

Address ....

....

Send to: Dept. YCM, RD Laboratories Ltd., 20 Court Road Estate, Cwmbran, Gwent NP44 3AS

RD Laboratories also supply the RD 8100 SYSTEM for economic automatic measurement, test and control. The modular-based RD 8100 SYSTEM is already used widely in many applications – from scientific experimentation to small-scale energy management systems. Please send stamp and address for brochure.

The high resolution colour graphics of the ZX Spectrum permit accurate presentation of complex or irregular images – maps, technical drawings, even personalities. But entering individual co-ordinates for unusual shapes can be tedious and time-consuming.

The RD DIGITAL TRACER cuts out tedious plotting. It provides instant transfer from original to diplay file – for screen display, ZX printer printout, or retention on cassette.

The RD DIGITAL TRACER is supplied with tracing sheet, software cassette and full instructions on use. Latest software includes fast colour fill, audible keystroke acknowledgement, fast PAPER colour change and fast LOAD from saved display file.

This is an invaluable tool for engineers, architects and other technicians, and for educational use in the presentation of lectures, and in computer training and application. Designed for the ZX Spectrum, the RD DIGITAL TRACER as supplied is compatible with ZX 81, although high resolution colour graphics are not available on this machine.

The RD DIGITAL TRACER is available from computer shops or direct from RD Laboratories. The direct (UK only) price of £55.50 includes Postage, Packing and full V.A.T. Send a cheque (payable to RD Laboratories Ltd.) with order for delivery within 28 days, or ask your local dealer for details.

R D Laboratories Ltd

20 Court Road Estate   Cwmbran   Gwent   NP44 3AS
(06333) 74333

ORIC SOFTWARE
Dept. YC(1) 118 Worcester Rd, Malvern, Worcs., WR14 1SS

SPECIALIST SUPPLIERS OF ORIC SOFTWARE
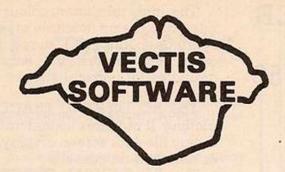
BIORHYTHMS 48K	5.95
SYNTHESIZER 48K	7.95
ZODIAC ADVENTURE 48K	9.95
CHESS 48K	9.95
ORIC BASE — Data base 48K	9.95
MULTI GAMES 48K	7.95
ORIC LANDER 48K	7.95
FORTH 48K	17.50

Postage & packing 50p (1 to 7 cassettes)

ORIC CASSETTE LEADS: — 7 PIN DIN — 3 JACKS		2.25
48K ORIC	NOW IN STOCK	169.95
	Postage and packing	5.95
CGP-115	4 COLOUR GRAPHIC PRINTER	149.00
	Cable for Oric Computer	22.95
	Postage & packing	5.95

Return to:
ORIC SOFTWARE, Dept YC(1), 118 Worcester Rd, Malvern, Worcs., WR14 1SS. Tele. Malvern (06845) 62467

I have a 16K Oric ☐	Mr/Mrs/Miss/Ms ....
48K Oric ☐	
My software interests are:—	Address ....
Home ☐ Business ☐	
Games ☐ Education ☐	....
Other (Please specify)	
	Postcode ....
	Cheque enclosed....................
	Please debit my Access/Visa
	AMex/Dclub — With £..............
	No. ....

SOFTWAREE WRITERS:— Lump sum or royalty payments. Send tapes & details.
(Tapes not returned unless postage is included).

ZX Workstations.. for Spectrum & ZX81

prices (£'s, inc VAT)
Spectrum 16
ZX81 15
postage 2
Extras:
power switch 3
alloy base 3.5
speaker 3.5

These stylish plinths, in tough moulded ABS, will raise and tilt the TV for better viewing, and angle the computer for easier typing. The power supply is fixed underneath. The printer can be used with the Spectrum version, the RAM with the '81', and the cassette player with both. Full details from: Peter Furlong Products, Unit 5c, South Coast Rd Ind. Estate, Peacehaven, Sussex BN9 8NA.   Tel (07914) 81637

FOX ELECTRONICS

NEW!

Presents Products:

For the Spectrum, The VIC-20, The Jupiter Ace and the ZX-81

The Fox **Spectrum Upgrade** Kit (for issue 2 machines only) takes your machine to 48K, no soldering required and full instructions supplied. **Unbeatable value at £24.00**

Improve your games with the Spectrum **Joystick.** Professionally built from strong nylon and steel. A must for all games lovers. ONLY **£25.00** + £1.00 P&P

New Sounds for your Spectrum with the *New* programmable sound generator. **£19.95** + 50p P&P. or inexpensively enhance existing sounds with a sound expansion for only £6.95 + 50p P&P.

Or let your Spectrum do the talking with the **Chatterbox** phonetic synthesis, unlimited vocabulary complete with speaker/-amplifier and housed in a smart black box. Also for **ZX81. Both at £56.00** incl. P&P.

New from FOX the **Vixen RAM Cartridge** for the VIC 20. Switchable between 16K or 8K + 3K gives you the option of full 16K RAM or 8K and 3K RAM in one pack. Simply plugs into rear expansion port no re-addressing of existing basic programs needed only **£39.95** incl. P&P.

Let your VIC 20 talk to you with the **New Chatterbox** speech synthesizer with an infinite vocabulary of spoken words. Fully programmable and simply plugs into VIC or Motherboard. Includes a series of software routine in Eprom to facilitate the programming. **£57.00** incl. P&P.

Coming shortly from FOX

THE TANDEM
Expandable expansion system, gives 4 expansion slots for VIC 20 cartridges. Custom designed case, plugs directly into computer. Further expanded by using Tandem system ROM socket. No extra power supply needed. **£33.00**

THE PACER
Uniquely expandable 16K RAM pack for the **Jupiter Ace.** Similar concept to ZX Panda. For more power to your **Faster Forth** you **need** a Pacer. 16K expandable RAM £29.95 16K Module £19.95 incl.

Improve your Memory with the ZX Panda. The **uniquely** expandable 16K RAM pack. Simply expanded to 32K by insertion of a plug in module. Specially designed to eliminate wobble, and complete with LED power indication
16K expandable RAM pack £19.95 incl. P&P.
16K expansion module £14.50 incl. P&P.
or 32K complete for £34.45 incl. P&P.
or go for the **Big Big** pack 64K only £50.00

ZX81 Keyboard (identical to Spectrum keyboard opposite again with ZX81 PCB fitting inside, again without soldering. A great buy at only **£29.95** + £2.50 P&P

ZX81 Keyboard No 2. Alternatively, go for the stylish and efficient, pushbutton, precision moulded (ABS as per your ZX81) 3 coloured keyboard at an incredibly low price of Unit adheres direct to ZX keyboard. No soldering required. **£10.00** incl. P&P.

Coming Shortly The Spectrum Echo. Volume and tone control. Load and Save without switching leads. Looks good, sounds good. **£23.50** incl. P&P.

Send now to FOX Electronics 141 Abbey Road, Popley 4. Basingstoke, Hants Catalogue available. Send SAE. All products guaranteed 1 yr. Overseas customers add a further £1.50 P&P to all items. Delivery 10-14 days.

Mind Bending or just good fun?

TOBOR
48K SPECTRUM
You are relentlessly pursued by Robots as you move through their city.
£7.95

JAWZ
16/48K SPECTRUM
Don scuba diving gear to shoot the killers of the deep.
£4.95
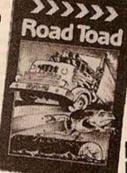
ROAD TOAD
16/48K SPECTRUM
You hop your Toad across the road.
£4.95

Elfin have created 3 new games for family enjoyment. There's TOBOR a test of skill and strategy, ROAD TOAD a test of reflex actions as you hop around the hazards and JAWZ to test your ability to fire at moving objects. All games are machine code to give fast arcade action and lively colourful graphics on your screen.

If you can't find them in your favourite store just fill in the coupon and post it to us with your remittance or phone your card number to us and we will see you have them by return post.

Trade enquiries. All games for 1 or 2 players.

Buy any 2 tapes and get the third for £1.25

ELFIN SOFTWARE More than fun and games.

Please send me:

Quantity		
	TOBOR £7.95	
	JAWZ £4.95	
	ROAD TOAD £4.95	

or use my credit card Access   Barclaycard

Card No. | | | | | | | | | | | |

Name

Address

To Elfin Software. Hudsons House, Battery Rd. Gt. Yarmouth NR30 3NN – Phone 0493-53170

Add 25p for postage & packing

YOUR COMPUTER CLASSIFIED

01-661 3036

Britain's Biggest Selling Home Computer Magazine

Weekend courses in Sinclair Basic, starting February 1983. 15 hours of instruction from a qualified lecturer, Friday evening to Sunday evening.
Luxurious 3-star Worcestershire Hotel, all rooms with bath and other facilities.
£55.00 inclusive.

GAINSBOROUGH HOUSE HOTEL
Bewdley Hill, Kidderminster
(0562) 754041

1/1

ZX81 — Cassette One — Fruit Machine, Hold, Nudge, Gamble, Respin, plus Pontoon — Cassette Two — Maze Adventure, Ambulance, Ambush — Cassette Three — Hangman, Caverns, Hi-Lo, Lunar Lander — £3.95 each. Morsoft, 39 Downing Drive, Morpeth NE61 2YB.

167/4

ZX 80/ 81 CASSETTE INTERFACE

Cassettes that were previously unloadable can now be loaded. Simplify your ZX80/81 program loading with your Interface Unit, our ZXi Interface Unit plugs neatly between the recorder and the ZX80/81. It filters and shapes the signal from the recorder and outputs a clean, constant level signal to the ZX80/81. It has an input level indicator, ZX Power on led and a protected ZX Reset Button. Connect up and raise the Recorder output until the led lamp lights, then 'Load' as normal.

Additionally, much faster 'saving' and 'loading' is possible, using a short machine code program.

Price: £17.00p
including VAT and postage.

If our ZXi Interface Unit does not simplify your program loading, please return it to us for a refund, within 14 days of purchase.

158/4

JELVALE LTD
104 Marshalswick Lane,
St Albans, Herts
Tel St Albans (0727) 69437

48K Spectrum. Two quality machine code programs, Reversi and Fruit Machine. £4.95. J. Reale, 23 Trowbridge Way, Kenton, Newcastle NE3 3TE.

170/4

CUBA. The revolutionary word game for two players. 16K ZX81. Cassette £4 from R. P. Wilson, 3 Homecroft Road, London SE26 5QN.

164/4

ZX81 16K Games, Scramble, Lander, Zap, Rescue £2.50 Richard Potter, 6 Warren Ridge, Frant, Kent.

166/4

ZX81 owners. Maze of Hilarious Horrors; One to four players great party game for the kids. Includes free game. £3.50 Collinson, 88 Sancroft Road, Derby.

171/4

48K Spectrum Screen Editor/Word Processor. Powerful screen facility, essential for microdrive users. £3.95. B P Mills 54 Petters Road, Ashtead, Surrey.

172/4

ATARI HOME COMPUTER CLUB: Good software/hardware at good discounts. (01) 647 1713 (after 5pm).

174/4

Pigeons 100 Competitors, Winners, Save 48K Spectrum £3. Alan Kirk, 13 Carey Park, Truro, Cornwall.

173/4

XERUS — Vic 20 (unexpanded); creative new fast action challenge, graduated skill levels, up to 80 independently moving aggressors, extensive colour/sound use, bit-mapped high resolution, machine code, excellent quality, cassette £7.85, Electronic Leisure, 11 Sherwood Close, Christchurch, Dorset BH23 2DD.

163/4

BROKEN YOUR TV/ COMPUTER JOYSTICK?

Inside your joystick handle is a white plastic insert. (The main cause of failure.) Stronger insert replacement now available. Supplied with full fitting instructions allowing you to be back in action in seconds.
Send £2.50/pair inc. p&p to: (YC)
Computer Supplies,
146 Church Road,
Boston,
Lincs. PE21 0JX

10/3

ZX81, leads, manual. 16K and DK'Tronics keyboard plus much software. Bargain £110 o.n.o. Phone 01 906 2849.

175/4

MZ-80A APPEND £3. Business, games, utility programs — catalogue SAE: DCS 38 South Parade, Bramhall, Stockport.

168/4

CBM 4016 and C2N including CBM manuals and books £400 o.n.o. Hunderford (0488) 84233 evenings.

165/4

ORIC Software: WE BUY. Send cassette (300 Bd) plus notes, for consideration, Nickelectric, 8 Old Pound Close, Lytchett Matravers, Dorset BH16 6BW.

161/4

MZ80K Dragon Crown 48K graphic adventure with M/C + squash, M/C Arcade favourite. Both for £5.00. Squirrelsoft, 33 Wilton Road, Edinburgh EH16.

MUSICSOFT
BBC MICRO 32K A/B.

5 music games of memory and dexterity. Multicolour graphics. Fun for 5 year olds yet frustrating for graduates. Variable speeds and scale levels. Cassette and Instructions £3.50

Cheques to Musicsoft
12 Fallowfield. Ampthill. Beds.

36/1

DRAGON 32 AND SHARP MZ80K SOFTWARE

Games Packs — £6.50 each.
Games Pack 1 — Bowling, UFO, Muncher, Micropoly, Mastermind.
Games Pack 2 — Race-chase, Depth Charge, Motor Cross, Glorious 12th, Canyon Bomber.
Games Pack 3 — Tank Battle, Reaction Time, Blind Maze, One Man and His Dog, Life.
Game Pack 4 — Poker, Pontoon, Roulette, Othello, Stock Broker.
(Sharp only)
Adventure Games
Devils Triangle — Sail to Bermuda but beware of the Bermuda Triangle! — £5.00.
Earth Rescue — Search the Galaxy for a rare mineral needed for Earths survival — £5.00.
King of the Valley — Try to control a kingdom and its people — £3.00.
Desert Patrol — Find your way across a desert but beware of its inhibitants! — £4.00.
Educational
Maths Tutor — A set of programmes designed to teach maths to children from 3-10 yrs. — £5.00.
Word Quiz — Find the hidden words in a jumble of letters — £3.00.
Send cheque/PO or SAE for details to
Abacus Software
20 Rhosleigh Ave.
Sharples Park
Bolton BL1 6PP.

BBC MODEL B
QUALITY SOFTWARE — KEEN PRICES

DOMGRID
Unravel the mystery pattern of dominoes created by the computer (a new problem every time).
Deceptively thought-provoking.
WORDGRID
The computer hides your words in the word square — then your opponent has to seek them out on line (no printer required).
Sensible, family games with educational value. £3.95 (inc) the pair.
SAE for list of these and other programs.
Strawberry Software
78a Sunderton Lane
Clanfield PORTSMOUTH
PO8 0NT

13/3

reprints

If you are interested in a particular article/special feature or advertisement in this journal
HAVE A GOOD LOOK AT OUR REPRINT SERVICE!
We offer an excellent, reasonably priced service working to your own specifications to produce a valuable and prestigeous addition to your promotional material. (Minimum order 250 copies). Telephone Michael Rogers on 01-661 3457 or complete and return the form below.

To: Michael Rogers, Your Computer, Reprint Department, Quadrant House, Sutton, Surrey SM2 5AS.

I am interested in . . . . copies of article/advert.

headed . . . . featured in this

journal on pages . . . . , issue dated . . . .

Please send me full details of your reprint service by return of post.

Name . . . .

Company . . . .

Address . . . .

. . . . Tel No . . . .

APPOINTMENTS

PROGRAMMERS WANTED

IJK Software Ltd., a leading name in **BBC Micro** software, are expanding their range to include software for Micros such as the **ORIC 1** and **LYNX**.
We pay top royalties or outright purchase for software, so don't just sit there – if you have written a programme for the **BBC Micro**, **ORIC 1 or LYNX**, and you want to be part of one of Britain's leading software houses, then send you program to us right away for appraisal in the strictest confidence.
Obviously, machine code arcade games will be the biggest money earners, but every program will be considered on its merits.
Our extensive nationwide advertising, together with national and international mail order system, dealer networks and distributors will ensure you get the best possible reward for your ingenuity.

Send to The Software Manager
IJK Software Ltd.
9 KING STREET, BLACKPOOL, LANCASHIRE.
TEL: 0253 21555

71/4

SELL IT WITH YOUR COMPUTER

CLASSIFIED ORDER FORM

Classified Rates
Linage: 30p per word (Min 15 words) — prepayable

Linage advertisers should complete the form provided in BLOCK CAPITALS. Phone number counts as 2 words. Name and address to be paid for if used in advertisement. Box Number if required is £3.00 extra.

Display — rates per scc (Min 2scc)

One Insertion	:	£7.00
Three Insertions	:	£6.70
Six Insertions	:	£6.50
Twelve Insertions	:	£6.00

Display advertisers should provide separate copy and preferably reserve space by phone (01-661 3036).

Method of Payment
Cheques etc, should be made payable to Business Press International Ltd, and crossed. I enclose herewith cheque/PO for

£..........................
Please debit my Access/Visa Barclay Card/American Express/Diners Club Int/ as below

SIGNATURE

Post to:
Cut out the order form and return together with your remittance to: Classified Department, Your Computer, Room H211, Quadrant House, The Quadrant, Sutton, Surrey SM2 5AS. (01) 661 3036

Please insert the following advertisement in Your Computer Classified Section

				LINAGE
				£4.50
				£6.00
				£7.50
				£9.00
				£10.50
				£12.00
				£13.50
				£15.00

No. of insertions required ☐ Box No. required YES/NO

NAME *(Please include initials)* ....

ADDRESS ....

....

....

Payment by credit card please state address card is registered

Daytime tel. no ....

THIS FORM SHOULD BE RETURNED BY MAY 3 FOR JUNE ISSUE PUBLICATION

Company Registered Number: 151537 (England) Registered Office: Quadrant House, The Quadrant, Sutton, Surrey SM2 5AS.